Quantifying and Improving Transferability in Domain Generalization

Guojun Zhang
School of Computer Science
University of Waterloo
Vector Institute
guojun.zhang@uwaterloo.ca

Han Zhao
Department of Computer Science
University of Illinois at Urbana-Champaign
hanzhao@illinois.edu

Yaoliang Yu **Pascal Poupart**
School of Computer Science
University of Waterloo
Vector Institute
{yaoliang.yu, ppoupart}@uwaterloo.ca

Abstract

Out-of-distribution generalization is one of the key challenges when transferring a model from the lab to the real world. Existing efforts mostly focus on building invariant features among source and target domains. Based on invariant features, a high-performing classifier on source domains could hopefully behave equally well on a target domain. In other words, the invariant features are *transferable*. However, in practice, there are no perfectly transferable features, and some algorithms seem to learn "more transferable" features than others. How can we understand and quantify such *transferability*? In this paper, we formally define transferability that one can quantify and compute in domain generalization. We point out the difference and connection with common discrepancy measures between domains, such as total variation and Wasserstein distance. We then prove that our transferability can be estimated with enough samples and give a new upper bound for the target error based on our transferability. Empirically, we evaluate the transferability of the feature embeddings learned by existing algorithms for domain generalization. Surprisingly, we find that many algorithms are not quite learning transferable features, although few could still survive. In light of this, we propose a new algorithm for learning transferable features and test it over various benchmark datasets, including RotatedMNIST, PACS, Office-Home and WILDS-FMoW. Experimental results show that the proposed algorithm achieves consistent improvement over many state-of-the-art algorithms, corroborating our theoretical findings.

1 Introduction

One of the cornerstone assumptions underlying the recent success of deep learning models is that the test data should share the same distribution as the training data. However, faced with ubiquitous distribution shifts in various real-world applications, such assumption hardly holds in practice. For example, a self-driving recognition system trained using data collected in the daytime may continually degrade its performance during nightfall. The system may also encounter weather or traffic conditions in a new city that never appear in the training set. In light of these potentially unseen scenarios, it is of paramount importance that the trained model can generalize *Out-Of-Distribution* (OOD): even if the target domain is not exactly the same as the source domain(s), the learned model should hopefully behave robustly under slight distribution shift.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

To this end, one line of works focuses on learning the so-called *invariant representations* [2, 16, 57, 58]. At a colloquial level, the goal here is to learn feature embeddings that lead to indistinguishable feature distributions from different domains. In practice, both the feature embeddings and the domain discriminators are often parametrized by neural networks, leading to an adversarial game between these two. Furthermore, in order to avoid degenerate solutions, the learned features are required to be informative about the output variable as well. This is enforced by placing a predictor over the features and minimize the corresponding supervised loss simultaneously [17, 32, 48, 49].

Another line of recent works aims to learn features that can induce *invariant predictors*, first termed as the invariant risk minimization (IRM) [3, 39] paradigm. Roughly speaking, the goal of IRM is to discover a feature embedding, upon which the optimal predictors, i.e., the Bayes predictor, are invariant across the training domains. Again, at the same time, the features should be informative about the output variable as well. However, the optimization problem of IRM is rather difficult, and several follow-up works have proposed different relaxations to the original formulation [1, 26].

Despite being extensively studied, both theoretical [41, 59] and empirical [20, 23] works have shown the insufficiency of existing algorithms for domain generalization (DG). Methods based on invariant features ignore the potential shift in the marginal label distributions across domains [59] and the methods based on invariant predictors are not robust to covariate shift [26]. Perhaps surprisingly, empirical works have shown that with proper data augmentation and careful model tuning, the very basic algorithm of empirical risk minimization (ERM) demonstrates superior performance on domain generalization over existing methods on benchmark image datasets [20, 23]. This sharp gap between theory and practice calls for a fundamental understanding of the following question:

> *What kind of invariance should we look for, in order to ensure that a good model on source domains also achieves decent accuracy on a related target domain?*

In this work we attempt to answer the above question by proposing a criterion for models to look at, dubbed as *transferability*, which asks for an invariance of the *excess risks* of a predictor across domains. Different from existing proposals of invariant features and invariant predictors, which seek to find feature embeddings that respectively induce invariant marginal and conditional distributions, our notion of transferability depends on the excess risk, hence it directly takes into account the joint distribution over both the features and the labels. We show how it can be used to naturally derive a new upper bound for the target error, and then we discuss how to estimate the transferability empirically with enough samples. Our definition also inspires a method that aims to find more transferable features via representation learning using adversarial training.

Empirically, we perform experiments to measure the transferability of several existing algorithms, on both small and large scale datasets. We show that many algorithms, including ERM, are not quite transferable under the definition (Fig. 1, see more details in §5): when we go away from the optimal classifier (with distance δ in the parameter space), it could happen that the source accuracy remains high but the target accuracy drops significantly. This implies that during the training process, an existing algorithm may find a good source classifier with low target accuracy, hence violating the requirement for invariance of excess risks. In contrast, our algorithm is more transferable, and achieves consistent improvement over existing state-of-the-art algorithms, corroborating our findings.

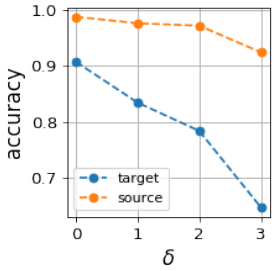

Figure 1: The target and source (test) accuracies of ERM on MNIST.

2 What is Transferability?

In this section we present our definition of transferability in the classification setting. The setup of domain generalization is the following:

Settings and Notations Given n labeled source domains $\mathcal{S}_1, \ldots, \mathcal{S}_n$, the problem of domain generalization is to learn a model from these source domains, in the hope that it performs well on an unseen target domain $\mathcal{T}$ that is "similar" to the source domains. Throughout the paper, we assume that both the source domains and the unseen target domain share the same input and output spaces, denoted as $\mathcal{X}$ and $\mathcal{Y}$, respectively. For multi-class classification, the output space $\mathcal{Y} = [K]$ is a set of

labels for multi-class classification. For binary classification, we consider $\mathcal{Y} = \{-1, +1\}$. Denote $\mathcal{H}$ as the hypothesis class. We define the classification error of a classifier $h \in \mathcal{H}$ on a domain $\mathcal{D}$ (or $\mathcal{S}$ for source domains, or $\mathcal{T}$ for target domains) as:[1]

$$\epsilon_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]. \tag{1}$$

For $\ell(h(x), y) = \mathbb{1}(h(x) \neq y)$, where $\mathbb{1}(\cdot)$ is the usual indicator function, we use $\epsilon_{\mathcal{D}}^{0-1}(h)$ to denote it is the 0-1 loss.

In domain generalization, we often have several source domains. For the ease of presentation, we only consider a single source domain in this section, and later extend to the general case in Section 5. Given two domains, the source domain $\mathcal{S}$ and the target domain $\mathcal{T}$, the task of domain generalization is to transfer a classifier h that performs well on $\mathcal{S}$ to $\mathcal{T}$. We ask: how much of the success of h on $\mathcal{S}$ can be transferred to $\mathcal{T}$?

Note that in order to evaluate the transferability from $\mathcal{S}$ to $\mathcal{T}$, we need information from the target domain, similar to the test phase in traditional supervised learning. We believe a good criterion of transferability should satisfy the following properties:

1. *Quantifiable:* the notion should be quantifiable and can be computed in practice;
2. Any near-optimal source classifier should be near-optimal on the target domain.
3. If the two domains are similar, as measured by e.g., total variation, then they are transferable to each other, but the converse may not be true.

At first glance the second criterion above might seem too strong and restrictive. However, we argue that in the task of domain generalization, we only have labeled source data and there is no clue to distinguish a classifier from another if both of them perform equally well on the source domain. Based on the second property, we first propose the following definition of transferability:

Definition 1 (transferability). *$\mathcal{S}$ is $(\delta_{\mathcal{S}}, \delta_{\mathcal{T}})_{\mathcal{H}}$-transferable to $\mathcal{T}$ if for $\delta_{\mathcal{S}} > 0$, there exists $\delta_{\mathcal{T}} > 0$ such that $\operatorname{argmin}(\epsilon_{\mathcal{S}}, \delta_{\mathcal{S}})_{\mathcal{H}} \subseteq \operatorname{argmin}(\epsilon_{\mathcal{T}}, \delta_{\mathcal{T}})_{\mathcal{H}}$, where:*

$$\operatorname{argmin}(\epsilon_{\mathcal{D}}, \delta)_{\mathcal{H}} := \{h \in \mathcal{H} : \epsilon_{\mathcal{D}}(h) \leq \inf_{h\in\mathcal{H}} \epsilon_{\mathcal{D}}(h) + \delta\}.$$

In the literature the set $\operatorname{argmin}(\epsilon_{\mathcal{D}}, \delta_{\mathcal{D}})_{\mathcal{H}}$ is also known as a *$\delta_{\mathcal{D}}$-minimal set* [24] of $\epsilon_{\mathcal{D}}$, which represents the near-optimal set of classifiers. Note that the δ-minimal set depends on the hypothesis class $\mathcal{H}$. Throughout the paper, we omit the subscript $\mathcal{H}$ in the definition when not causing confusion. Def. 1 says that near-optimal source classifiers are also near-optimal target classifiers. Furthermore, our transferability is transitive: if $\mathcal{S}$ is $(\delta_{\mathcal{S}}, \delta_{\mathcal{P}})$-transferable to $\mathcal{P}$, and $\mathcal{P}$ is $(\delta_{\mathcal{P}}, \delta_{\mathcal{T}})$-transferable to $\mathcal{T}$, then $\mathcal{S}$ is $(\delta_{\mathcal{S}}, \delta_{\mathcal{T}})$-transferable to $\mathcal{T}$.

Next we define transfer measures, which we will show to be equivalent with Def. 1 in Prop. 5.

Definition 2 (quantifiable transfer measures). *Given some $\Gamma \subseteq \mathcal{H}$, $\epsilon_{\mathcal{S}}^* := \inf_{h\in\Gamma} \epsilon_{\mathcal{S}}(h)$ and $\epsilon_{\mathcal{T}}^* := \inf_{h\in\Gamma} \epsilon_{\mathcal{T}}(h)$ we define the one-sided transfer measure, symmetric transfer measure and the realizable transfer measure respectively as:*

$$\mathtt{T}_{\Gamma}(\mathcal{S}\|\mathcal{T}) := \sup_{h\in\Gamma} \epsilon_{\mathcal{T}}(h) - \epsilon_{\mathcal{T}}^* - (\epsilon_{\mathcal{S}}(h) - \epsilon_{\mathcal{S}}^*), \tag{2}$$

$$\mathtt{T}_{\Gamma}(\mathcal{S}, \mathcal{T}) := \max\{\mathtt{T}_{\Gamma}(\mathcal{S}\|\mathcal{T}), \mathtt{T}_{\Gamma}(\mathcal{T}\|\mathcal{S})\} = \sup_{h\in\Gamma} |\epsilon_{\mathcal{S}}(h) - \epsilon_{\mathcal{S}}^* - (\epsilon_{\mathcal{T}}(h) - \epsilon_{\mathcal{T}}^*)|, \tag{3}$$

$$\mathtt{T}_{\Gamma}^{\mathtt{r}}(\mathcal{S}, \mathcal{T}) := \sup_{h\in\Gamma} |\epsilon_{\mathcal{S}}(h) - \epsilon_{\mathcal{T}}(h)|. \tag{4}$$

The distinction between Γ and $\mathcal{H}$ will become apparent in Prop. 5. Note that the one-sided transfer measure is not symmetric. If we want the two domains $\mathcal{S}$ and $\mathcal{T}$ to be mutually transferable to each other, we can use the symmetric transfer measure. We call both quantities as *transfer measures*. Furthermore, the symmetric transfer measure reduces to (4) in the realizable case when $\epsilon_{\mathcal{S}}^* = \epsilon_{\mathcal{T}}^* = 0$. In statistical learning theory, $\epsilon_{\mathcal{D}}(h) - \epsilon_{\mathcal{D}}^*$ is often known as an *excess risk* [24], which is the relative error compared to the optimal classifier. The transfer measures can thus be represented with the difference of excess risks. With Def. 2, we can immediately obtain the following result that upper bounds the target error:

[1] Throughout the paper, we will use the terms domain and distribution interchangeably.

Proposition 3 (**target error bound**). *Given $\Gamma \subseteq \mathcal{H}$, for any $h \in \Gamma$, the target error is bounded by:*

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h) + \epsilon^*_{\mathcal{T}} - \epsilon^*_{\mathcal{S}} + \mathtt{T}_\Gamma(\mathcal{S}\|\mathcal{T}) \leq \epsilon_{\mathcal{S}}(h) + \epsilon^*_{\mathcal{T}} - \epsilon^*_{\mathcal{S}} + \mathtt{T}_\Gamma(\mathcal{S},\mathcal{T}). \tag{5}$$

The first error bound of such type for a target domain uses $\mathcal{H}$-divergence [8, 9, 12] for binary classification (or more rigorously, $\mathcal{H}\Delta\mathcal{H}$-divergence). The main difference between ours and $\mathcal{H}$-divergence is that $\mathcal{H}$-divergence only concerns about the marginal input distributions, whereas the transfer measures depend on the joint distributions over both the inputs and the labels. We note that Proposition 3 is general and works in the multi-class case as well. Moreover, even in the binary classification case we can prove that our Proposition 3 is tighter than $\mathcal{H}$-divergence (see Proposition 27).

In practice we may not know the optimal errors. In this case, we can use the realizable transfer measure to upper bound the symmetric transfer measure (note that $\epsilon^*_{\mathcal{S}}$ or $\epsilon^*_{\mathcal{T}}$ may not be zero):

Proposition 4. *For $\Gamma \subseteq \mathcal{H}$ and domains $\mathcal{S}$, $\mathcal{T}$ we have: $\mathtt{T}_\Gamma(\mathcal{S},\mathcal{T}) \leq 2\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T})$.*

Since Def. 1 essentially asks that the excess risks of approximately optimal classifiers on the source domain are comparable between the source and target domains, we can show that Def. 1 and Def. 2 are equivalent if Γ is a δ-minimal set:

Proposition 5 (**equivalence between transferability and transfer measures**). *Let $\delta_{\mathcal{S}} > 0$ and $\Gamma = \operatorname{argmin}(\epsilon_{\mathcal{S}}, \delta_{\mathcal{S}})$ and suppose $\inf_{h\in\Gamma} \epsilon_{\mathcal{T}}(h) = \inf_{h\in\mathcal{H}} \epsilon_{\mathcal{T}}(h)$. If $\mathtt{T}_\Gamma(\mathcal{S}\|\mathcal{T}) \leq \delta$ or $\mathtt{T}_\Gamma(\mathcal{S},\mathcal{T}) \leq \delta$, then $\mathcal{S}$ is $(\delta_{\mathcal{S}}, \delta + \delta_{\mathcal{S}})$-transferable to $\mathcal{T}$. If $\mathcal{S}$ is $(\delta_{\mathcal{S}}, \delta_{\mathcal{T}})$-transferable to $\mathcal{T}$, then $\mathtt{T}_\Gamma(\mathcal{S}\|\mathcal{T}) \leq \delta_{\mathcal{T}}$ and $\mathtt{T}_\Gamma(\mathcal{S},\mathcal{T}) \leq \max\{\delta_{\mathcal{S}}, \delta_{\mathcal{T}}\}$.*

In Prop. 5, we do not require $\Gamma = \mathcal{H}$ since it is unnecessary to impose that all classifiers in $\mathcal{H}$ have similar excess risks on source and target domains. Instead, we only constrain Γ to be a δ-minimal set, i.e., Γ includes approximately optimal classifiers of $\mathcal{S}$. See also Example 8. An additional assumption is that Γ also includes the optimal classifier of $\mathcal{T}$ which can be ensured by controlling $\delta_{\mathcal{S}}$.

2.1 Comparison with other discrepancy measures between domains

In this subsection, we compare the realizable transfer measure (4) with other discrepancy measures between domains and focus on the 0-1 loss $\epsilon^{0-1}_{\mathcal{D}}$. We first note that $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T})$ can be written as an integral probability metric (IPM) [36, 46]. The l.h.s. of (4) can be written as:

$$\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T}) := d_{\mathcal{F}_\Gamma}(\mathcal{S},\mathcal{T}), \text{ where } d_{\mathcal{F}}(\mathcal{S},\mathcal{T}) = \sup_{f\in\mathcal{F}} \left| \sum_y \int f(x,y)(p_{\mathcal{S}}(x,y) - p_{\mathcal{T}}(x,y))dx \right|, \tag{6}$$

and $\mathcal{F}_\Gamma := \{(x,y) \mapsto \mathbb{1}(h(x) \neq y), h \in \Gamma\}$. Typical IPMs [46] include MMD, Wasserstein distance, Dudley metric and the Kolmogorov–Smirnov distance (see Appendix B.2 for more details). However, $\mathcal{F}_\Gamma$ is fundamentally different from these IPMs since it relies on an underlying function class Γ. Our realizable transfer measure shares some similarity with Arora et al. [4], where a changeable function class is used, but the exact choices of the function class are different.

Even though the transferability can be written in terms of IPM, it is in fact a pseudo-metric:

Proposition 6 (**pseudo-metric**). *For a general loss $\epsilon_{\mathcal{D}}$ as in (1), $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T})$ is a pseudo-metric, i.e., for any distributions $\mathcal{S},\mathcal{T},\mathcal{P}$ on the same underlying space, we have $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{S}) = 0$, $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T}) = \mathcal{T}^{\mathtt{r}}_\Gamma(\mathcal{T},\mathcal{S})$ (symmetry), and $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T}) \leq \mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{P}) + \mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{P},\mathcal{T})$ (triangle inequality).*

However in general $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T})$ is not a metric since $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T}) = 0$ even if $\mathcal{S} \neq \mathcal{T}$. For instance, taking $\Gamma = \{h^*\}$ to be the optimal classifier on both $\mathcal{S}$ and $\mathcal{T}$. we have $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T}) = 0$, but $\mathcal{S}$ and $\mathcal{T}$ could differ a lot (see Figure 2). In the next result we discuss the connection between realizable transfer measures and total variation (c.f. Appendix B.2).

Proposition 7 (**equivalence with total variation**). *For binary classification with labels $\{-1, 1\}$, given the 0-1 loss $\epsilon_{\mathcal{D}} = \epsilon^{0-1}_{\mathcal{D}}$, we have $\mathtt{T}^{\mathtt{r}}_\Gamma(\mathcal{S},\mathcal{T}) \leq d_{\mathrm{TV}}(\mathcal{S},\mathcal{T})$ for domains $\mathcal{S},\mathcal{T}$ and any $\Gamma \subseteq \mathcal{H}$. Denote $\mathcal{H}_t$ to be the set of all binary classifiers. Then we have $d_{\mathrm{TV}}(\mathcal{S},\mathcal{T}) \leq 4\mathtt{T}^{\mathtt{r}}_{\mathcal{H}_t}(\mathcal{S},\mathcal{T})$.*

Prop. 7 tells us that transfer measures (see also Prop. 4) are no stronger than total variation, and in the realizable case, (3) is equivalent to the similarity of domains (as measured by total variation) if Γ is unconstrained. We can moreover show that transfer measures are strictly weaker, if we choose Γ to be some δ-minimal set:

Example 8 (**very dissimilar joint distributions but transferable**). *We study the distributions described in Figure 2. The joint distributions are very dissimilar, i.e., for any X, Y in the domain, $|p_{\mathcal{S}}(X,Y) - p_{\mathcal{T}}(X,Y)| = 0.8$. Define*

$$h_\rho(X) = \begin{cases} 1 & \text{if } -1 \le X < \rho \\ -1 & \text{if } \rho \le X < 1 \end{cases}. \tag{7}$$

We choose the hypothesis class $\mathcal{H} = \{h_\rho,\ \rho \in [-1,1]\}$ and $\Gamma = \{h_\rho, |\rho| \le \delta/0.8\}$ (for small δ, say $\delta < 0.01$) to be some neighborhood of the optimal source classifier $h^* = h_0$. Then $\mathtt{T}_\Gamma(\mathcal{S},\mathcal{T}) = \sup_{h\in\Gamma} |\epsilon_{\mathcal{S}}(h) - \epsilon_{\mathcal{T}}(h)| = \delta$, and $\mathcal{S}$ is $(\delta_{\mathcal{S}}, \delta+\delta_{\mathcal{S}})$-transferable to $\mathcal{T}$ on Γ for any $\delta_{\mathcal{S}} > 0$ according to Prop. 5. Note that $\epsilon^*_{\mathcal{S}} = \epsilon^*_{\mathcal{T}} = 0$.

3 Computing Transferability

In the last section we proposed a new concept called transferability. However, although Def. 1 provides a theoretically sound result for transferability, it is hard to verify it in practice, since we cannot exhaust all approximately good classifiers, especially for rich models such as deep neural networks. Nevertheless, Prop. 3 and Prop. 5 provide a framework to compute transferability through transfer measures, despite their simplicity. In this section we discuss how to compute these quantities by making necessary approximations based on transfer measures. There are two difficulties we need to overcome: **(1)** In practice we only have finite samples drawn from true distributions; **(2)** We need a surrogate loss such as cross entropy for training and the 0-1 loss for evaluation. In §3.1 we show that our transfer measures can be estimated with enough samples, and in §3.2 we discuss transferability with a surrogate loss. These results will be used in our algorithms in the next section.

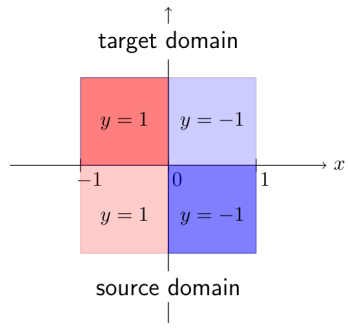

Figure 2: Visualization of Example 8. **Source domain**: $P_{\mathcal{S}}(Y=1, -1 \le X < 0) = 0.1$, $P_{\mathcal{S}}(Y=-1, 0 \le X < 1) = 0.9$. **Target domain**: $P_{\mathcal{T}}(Y=1, -1 \le X < 0) = 0.9$, $P_{\mathcal{T}}(Y=-1, 0 \le X < 1) = 0.1$. The dark and light colors show the intensity of the probability mass. The vertical axis denotes whether it is the target or source domain (above or below x-axis).

3.1 Estimation of transferability

We show how to estimate the transfer measure $\mathtt{T}_\Gamma(\mathcal{S}\|\mathcal{T})$ from finite samples. Other versions of transfer measures in Def. 1 follow analogously (see Appendix A for more details).

Lemma 9 (**reduction of estimation error**). *Given general loss $\epsilon_{\mathcal{D}}$ as in* (1)*, suppose $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{T}}$ are i.i.d. sample distributions drawn from distributions of $\mathcal{S}$ and $\mathcal{T}$, then for any $\Gamma \subseteq \mathcal{H}$ we have:*

$$\mathtt{T}_\Gamma(\mathcal{S}\|\mathcal{T}) \le \mathtt{T}_\Gamma(\widehat{\mathcal{S}}\|\widehat{\mathcal{T}}) + 2\mathrm{est}_\Gamma(\mathcal{S}) + 2\mathrm{est}_\Gamma(\mathcal{T}),$$

with the estimation errors $\mathrm{est}_\Gamma(\mathcal{S}) = \sup_{h\in\Gamma} |\epsilon_{\mathcal{S}}(h) - \epsilon_{\widehat{\mathcal{S}}}(h)|$, $\mathrm{est}_\Gamma(\mathcal{T}) = \sup_{h\in\Gamma} |\epsilon_{\mathcal{T}}(h) - \epsilon_{\widehat{\mathcal{T}}}(h)|$.

This lemma tells us that estimating transferability is no harder than computing the estimation errors of both domains. If the function class Γ has uniform convergence property [44], then we can guarantee efficient estimation of transferability. We first bound the sample complexity through Rademacher complexity, which is a standard tool in bounding estimation errors [5]:

Theorem 10 (**estimation error with Rademacher complexity**). *Given the 0-1 loss $\epsilon_{\mathcal{D}} = \epsilon^{0-1}_{\mathcal{D}}$, suppose $\widehat{\mathcal{S}}$ and $\widehat{\mathcal{T}}$ are sample sets with m and k samples drawn i.i.d. from distributions $\mathcal{S}$ and $\mathcal{T}$, respectively. For any $\Gamma \subseteq \mathcal{H}$ the following holds with probability $1-\delta$:*

$$\mathtt{T}_\Gamma(\mathcal{S}\|\mathcal{T}) \le \mathtt{T}_\Gamma(\widehat{\mathcal{S}}\|\widehat{\mathcal{T}}) + 4\mathfrak{R}_m(\mathcal{F}_\Gamma) + 4\mathfrak{R}_k(\mathcal{F}_\Gamma) + 2\sqrt{\frac{\log(4/\delta)}{2m}} + 2\sqrt{\frac{\log(4/\delta)}{2k}},$$

where $\mathcal{F}_\Gamma := \{(x,y) \mapsto \mathbb{1}(h(x) \ne y),\ h \in \Gamma\}$. If furthermore, Γ is a set of binary classifiers with labels $\{-1,1\}$, then $2\mathfrak{R}_m(\mathcal{F}_\Gamma) = \mathfrak{R}_m(\Gamma)$, $2\mathfrak{R}_k(\mathcal{F}_\Gamma) = \mathfrak{R}_k(\Gamma)$.

We also provide estimation error results using Vapnik–Chervonenkis (VC) dimension and Natarajan dimension in Appendix B.3. It is worth mentioning that the VC dimension of piecewise-polynomial neural networks has been upper bounded in Bartlett et al. [7]. Since transfer measures can be estimated, in later sections we do not distinguish the sample sets $\widehat{\mathcal{S}}, \widehat{\mathcal{T}}$ and the underlying distributions $\mathcal{S}, \mathcal{T}$.

3.2 Transferability with a surrogate loss

Due to the intractability of minimizing the 0-1 loss, we need to use a surrogate loss [6] for training in practice. In this section, we discuss this nuance w.r.t. transferability. We will focus on the most commonly used surrogate loss, cross entropy (CE), although some of the results can be easily adapted to other loss functions. To distinguish a surrogate loss from the 0-1 loss, we use $\epsilon_{\mathcal{D}}$ from now on for a surrogate loss and $\epsilon_{\mathcal{D}}^{0-1}$ for the 0-1 loss. One of the difficulties is the non-equivalence between δ-minimal sets w.r.t. the 0-1 loss and a surrogate loss, i.e. $\operatorname{argmin}(\epsilon_{\mathcal{D}}, \delta)$ might be quite different from $\operatorname{argmin}(\epsilon_{\mathcal{D}}^{0-1}, \delta)$. Moreover, it is not practical to find all elements in $\operatorname{argmin}(\epsilon_{\mathcal{D}}^{0-1}, \delta)$ since the loss is nonconvex and nonsmooth. In light of these difficulties, we propose a more practical notion of transferability based on surrogate loss $\epsilon_{\mathcal{D}}$:

Proposition 11 (**transfer measure with a surrogate loss**). *Given surrogate loss $\epsilon_{\mathcal{D}} \geq \epsilon_{\mathcal{D}}^{0-1}$ on a general domain $\mathcal{D}$. Suppose $\Gamma = \operatorname{argmin}(\epsilon_{\mathcal{S}}, \delta_{\mathcal{S}})$ and denote $\epsilon_{\mathcal{T}}^* = \inf_{h\in\Gamma} \epsilon_{\mathcal{T}}(h)$, $\epsilon_{\mathcal{S}}^* = \inf_{h\in\Gamma} \epsilon_{\mathcal{S}}(h)$, $(\epsilon_{\mathcal{T}}^{0-1})^* = \inf_{h\in\mathcal{H}} \epsilon_{\mathcal{T}}^{0-1}(h)$. If the following holds:*

$$\mathtt{T}_{\Gamma}(\mathcal{S}\|\mathcal{T}) = \sup_{h\in\Gamma} \epsilon_{\mathcal{T}}(h) - \epsilon_{\mathcal{T}}^* - (\epsilon_{\mathcal{S}}(h) - \epsilon_{\mathcal{S}}^*) \leq \delta, \tag{8}$$

then we have $\operatorname{argmin}(\epsilon_{\mathcal{S}}, \delta_{\mathcal{S}}) \subseteq \operatorname{argmin}(\epsilon_{\mathcal{T}}^{0-1}, \delta + \delta_{\mathcal{S}} + \epsilon_{\mathcal{T}}^* - (\epsilon_{\mathcal{T}}^{0-1})^*)$.

This proposition implies that if the transfer measure is small, then a near-optimal classifier of the surrogate loss in the source domain would be near-optimal in the target domain for the 0-1 loss. It also gives us a practical framework to guarantee transferability, which we will discuss in more depth in Section 4. Assume $\epsilon_{\mathcal{D}} : \mathcal{H} \to \mathbb{R}$ to be Lipschitz continuous and strongly convex, which is satisfied for the cross entropy loss (see Appendix B.4). We are able to translate the δ-minimal set to L_p balls in the function space:

$$C_1\|h - h^*\|_{2,\mathcal{D}} \leq \epsilon_{\mathcal{D}}(h) - \epsilon_{\mathcal{D}}(h^*) \leq C_2\|h - h^*\|_{1,\mathcal{D}}, \tag{9}$$

where C_1 and C_2 are absolute constants and h^* is an optimal classifier. The function norms $\|\cdot\|_{1,\mathcal{D}}$ and $\|\cdot\|_{2,\mathcal{D}}$ are the usual L_p norms over distribution $\mathcal{D}$. Since the classifier $h = q(\theta, \cdot)$ is usually parameterized with, say a neural network, we further upper bound the function norms by the distance of parameters, that is, for $1 \leq p < \infty$, $h = q(\theta, \cdot)$ and $h' = q(\theta', \cdot)$, we have $\|h - h'\|_{p,\mathcal{D}} \leq L\|\theta - \theta'\|_2$, with L some Lipschitz constant of q (Appendix B.4). Combined with (9), we obtain:

$$\epsilon_{\mathcal{D}}(h) - \epsilon_{\mathcal{D}}(h') \leq LC_2\|\theta - \theta'\|_2. \tag{10}$$

In other words, if the parameters are close enough, then the losses should not differ too much. We denote $\|\cdot\|_2$ as the Euclidean norm, and for later convenience we will omit the subscript in $\|\cdot\|_2$.

4 Algorithms for Evaluating and Improving Transferability

The notion of transferability is defined w.r.t. domains, hence by learning feature embeddings that induce certain feature distributions, one can aim to improve transferability of two given domains. In this section we design algorithms to evaluate and improve transferability by learning such transformations. To start with, let $g : \mathcal{X} \to \mathcal{Z}$ be a feature embedding (a.k.a. featurizer), where $\mathcal{Z}$ is understood to be a feature space. By a joint distribution $\mathcal{D}^g$ (or $\mathcal{S}^g$, $\mathcal{T}^g$) we mean a distribution on $g(\mathcal{X}) \times \mathcal{Y}$. Formally, we are dealing with push-forwards of distributions:

$$\mathcal{S}^g := (g, \mathrm{id})\#\mathcal{S},\ \mathcal{T}^g := (g, \mathrm{id})\#\mathcal{T}, \tag{11}$$

where $(g, \mathrm{id}) : (x, y) \mapsto (g(x), y)$ is a function on $\mathcal{X} \times \mathcal{Y}$. $\mathcal{S}$ and $\mathcal{T}$ here are joint distributions on $\mathcal{X} \times \mathcal{Y}$, and here we specify $\mathcal{X}$ to be the space of the original signal such as an image. Since $\mathcal{S}$ and $\mathcal{T}$ cannot be changed, what we are evaluating here is the feature embedding g. The key quantity is transfer measures as in (8):

$$\mathtt{T}_{\Gamma}(\mathcal{S}^g\|\mathcal{T}^g) = \sup_{h\in\Gamma} \epsilon_{\mathcal{T}^g}(h) - \epsilon_{\mathcal{T}^g}^* - (\epsilon_{\mathcal{S}^g}(h) - \epsilon_{\mathcal{S}^g}^*), \quad \Gamma = \operatorname{argmin}(\epsilon_{\mathcal{S}^g}, \delta_{\mathcal{S}^g}). \tag{12}$$

Although Γ is hard to compute, we can use (10) to obtain a lower bound of (12). That is, given a parametrization of the classifier $h = q(\theta, \cdot)$ and the optimal classifier $h^* = q(\theta^*, \cdot)$, we have:

$$\begin{aligned}
\mathtt{T}_\Gamma(\mathcal{S}^g \| \mathcal{T}^g) &\geq \sup_{\|\theta-\theta^*\|\leq\delta} \epsilon_{\mathcal{T}^g}(h) - \epsilon_{\mathcal{S}^g}(h) - \epsilon^*_{\mathcal{T}^g} + \epsilon^*_{\mathcal{S}^g} \\
&\geq \sup_{\|\theta-\theta^*\|\leq\delta} \epsilon_{\mathcal{T}^g}(h) - \epsilon_{\mathcal{S}^g}(h) - \epsilon_{\mathcal{T}^g}(\widehat{h^*}) \\
&\approx \sup_{\|\theta-\widehat{\theta^*}\|\leq\delta} \epsilon_{\mathcal{T}^g}(h) - \epsilon_{\mathcal{S}^g}(h) - \epsilon_{\mathcal{T}^g}(\widehat{h^*}) \qquad (13)
\end{aligned}$$

where $\delta > 0$ depends on Γ and the constant in (10). In the second and the third lines, we approximated the optimal errors $\epsilon^*_{\mathcal{T}^g}$ and $\epsilon^*_{\mathcal{S}^g}$ with $0 \leq \epsilon^*_{\mathcal{S}^g} \leq \epsilon_{\mathcal{S}^g}(\widehat{h^*})$, $0 \leq \epsilon^*_{\mathcal{T}^g} \leq \epsilon_{\mathcal{T}^g}(\widehat{h^*})$, and we use the learned classifier $\widehat{h^*} = q(\widehat{\theta^*}, \cdot)$ as a surrogate for the optimal classifier. As a result, if the r.h.s. of (13) is large, then $\mathcal{S}^g$ is not quite transferable to $\mathcal{T}^g$.

Algorithm 1: Algorithm for evaluating transferability among multiple domains

Input: learned feature embedding g, learned classifier $\widehat{h^*} = q(\widehat{\theta^*}, \cdot)$, target sample training set $\mathcal{T} = \mathcal{S}_0$, sample training sets $\mathcal{S}_1, \ldots, \mathcal{S}_n$, ascent optimizer, minimal errors $\epsilon^*_{\mathcal{S}_i} \approx \epsilon_{\mathcal{S}_i}(\widehat{h^*})$, adversarial radius δ

Initialize: a classifier $h = q(\theta, \cdot)$ and $\theta = \widehat{\theta^*}$, gap $= -\infty$

for t *in* $1 \ldots T$ **do**

- Find $\max_i \epsilon_{\mathcal{S}_i}(h \circ g)$ and $\min_i \epsilon_{\mathcal{S}_i}(h \circ g)$ and corresponding indices j and k
- Run an ascent optimizer on h to maximize $\mathrm{gap}_0 = \epsilon_{\mathcal{S}_j}(h \circ g) - \epsilon_{\mathcal{S}_k}(h \circ g)$
- Project θ onto the Euclidean ball $\|\theta - \widehat{\theta^*}\| \leq \delta$
- **if** $\mathrm{gap}_0 > \mathrm{gap}$ **then**
  - $\mathrm{gap} = \mathrm{gap}_0$, save accuracies and losses of each domain

Output: j, k, h, $\epsilon_{\mathcal{S}_j}(h \circ g) - \epsilon_{\mathcal{S}_k}(h \circ g)$, $\epsilon_{\mathcal{S}_j}(\widehat{h^*})$, $\epsilon_{\mathcal{S}_k}(\widehat{h^*})$

We can thus design an algorithm to evaluate the transferability in Section 4.1. By computing the lower bound in (13), we can disprove the transferability as in Prop. 5 and Prop. 11. Computing the lower bound in (13) can be regarded as an attack method: there is an adversary trying to show that $\mathcal{S}^g$ is not transferable to $\mathcal{T}^g$. For this attack, we could also design a defence method aiming to minimize the lower bound and learn more transferable features.

4.1 Algorithm for evaluating transferability

In domain generalization we have one target domain and more than one source domains. To ease the presentation, we denote $\mathcal{S}_0 = \mathcal{T}$ (and thus $\mathcal{S}^g_0 = \mathcal{T}^g$) and extend the index set to be $\{0, 1, \cdots, n\}$. We need to evaluate the transferability (13) between all pairs of $\mathcal{S}^g_i$ and $\mathcal{S}^g_j$. Algorithm 1 gives an efficient method to compute the worst-case gap $\sup_{\|\theta-\widehat{\theta^*}\|\leq\delta} \epsilon_{\mathcal{S}^g_i}(h) - \epsilon_{\mathcal{S}^g_j}(h)$ among all pairs of (i, j). Essentially, it finds the worst pair of (i, j) at each step such that the gap $\epsilon_{\mathcal{S}^g_i}(h) - \epsilon_{\mathcal{S}^g_j}(h)$ takes the largest value, and then maximize this gap over parameter θ through gradient ascent.

Note that the computation of (13) also depends on the information from the target domain. This is valid since we are only *evaluating* but not *training* over these domains.

4.2 Algorithm for improving transferability

The evaluation sub-procedure provides us a way to pick a pair of non-transferable domains $(\mathcal{S}^g_i, \mathcal{S}^g_j)$, which in turn could be used to improve the transferability among all source domains by updating the feature embedding g such that the gap $\sup_{\|\theta-\theta^*\|\leq\delta} \epsilon_{\mathcal{S}^g_i}(h) - \epsilon_{\mathcal{S}^g_j}(h)$ for $(i, j) \in [n] \times [n]$. Simultaneously, we also require that the feature embedding g preserves information for the target task of interest. With the parametrization $h = q(\theta, \cdot)$, $h' = q(\theta', \cdot)$, the overall optimization problem

can be formulated as:

$$\min_{g,h} \max_{\|\theta'-\theta\|\le\delta} \frac{1}{n}\sum_{i=1}^{n} \epsilon_{\mathcal{S}_i}(h\circ g) + \left(\max_i \epsilon_{\mathcal{S}_i}(h'\circ g) - \min_i \epsilon_{\mathcal{S}_i}(h'\circ g)\right). \tag{14}$$

Intuitively, we want to learn a common feature embedding and a classifier such that all source errors are small and the pairwise transferability between source domains is also small. If the optimization problem is properly solved, then we have the following guarantee:

Theorem 12 (**optimization guarantee**). *Assume that the function $q(\cdot, x)$ is L_θ Lipschitz continuous for any x. Suppose we have learned a feature embedding g and a classifier h such that the loss functional $\epsilon_{\mathcal{S}_i^g} : \mathcal{H} \to \mathbb{R}$ is L_ℓ Lipschitz continuous w.r.t. distribution $\mathcal{S}_i^g$ for $i \in [n]$ and*

$$\max_{\|\theta'-\theta\|\le\delta} \frac{1}{n}\sum_{i=1}^{n} \epsilon_{\mathcal{S}_i}(h\circ g) + \left(\max_i \epsilon_{\mathcal{S}_i}(h'\circ g) - \min_i \epsilon_{\mathcal{S}_i}(h'\circ g)\right) \le \eta, \tag{15}$$

where θ, θ' are parameters of h and h'. Then for any $h' \in \Gamma = \{q(\theta', \cdot) : \|\theta - \theta'\| \le \delta\}$, we have:

$$\mathtt{T}^{\mathtt{r}}_{\Gamma}(\mathcal{T}_1^g, \mathcal{T}_2^g) \le \eta, \quad \epsilon_{\mathcal{S}_i}(h'\circ g) \le \eta + L_\ell L_\theta \delta, \quad \epsilon_{\mathcal{T}}(h'\circ g) \le 2\eta + L_\ell L_\theta \delta, \tag{16}$$

for any $\mathcal{T}_1^g, \mathcal{T}_2^g, \mathcal{T}^g \in \mathrm{conv}(\mathcal{S}_1^g, \ldots, \mathcal{S}_n^g)$ and any $i \in [n]$.

The Lipschitzness assumption for $\epsilon_{\mathcal{S}_i^g}$ is mild and can be satisfied for cross entropy loss (c.f. Appendix B.4.1). Here $\mathrm{conv}(\cdot)$ denotes the convex hull in the same sense as Albuquerque et al. [2], i.e., each element is a mixture of source distributions. Thm 12 tells us that if we can solve the optimization problem (14) properly, we can guarantee transferability on a neighborhood of the classifier, as an approximation of the δ-minimal set. We thus propose Algorithm 2, which shares similarity with existing frameworks, such as DANN [16] and Distributional Robust Optimization [43, 45], in the sense that they all involve adversarial training and minimax optimization. However, the objective in our case is different and we provide a more detailed comparison with existing methods in Appendix B.5.

Algorithm 2: Transfer algorithm for domain generalization

Input: samples sets of source domains $\mathcal{S}_1, \ldots, \mathcal{S}_n$, feature embedding g, classifier $h = q(\theta, \cdot)$, adversarial classifier $h' = q(\theta', \cdot)$, surrogate loss $\epsilon_{\mathcal{D}}$, adversarial radius δ, ascent optimizer, descent optimizer, weight parameter λ, number of epochs T

for t *in* $1 \ldots T$ **do**
- Compute $\max_i \epsilon_{\mathcal{S}_i}(h \circ g)$ and $\min_i \epsilon_{\mathcal{S}_i}(h \circ g)$
- Initialization $h' = h$ (or $\theta' = \theta$)
- **for** k *in* $1 \ldots N$ **do**
  - Run the ascent optimizer on h' to maximize $\max_i \epsilon_{\mathcal{S}_i}(h' \circ g) - \min_i \epsilon_{\mathcal{S}_i}(h' \circ g)$ fixing g
  - Project θ' onto the Euclidean ball $\|\theta' - \theta\| \le \delta$
- Fixing h', run the descent optimizer on g, h to minimize error $= \frac{1}{n}\sum_i \epsilon_{\mathcal{S}_i}(h \circ g) + (\max_i \epsilon_{\mathcal{S}_i}(h' \circ g) - \min_i \epsilon_{\mathcal{S}_i}(h' \circ g))$

Output: feature embedding g, classifier h

5 Experiments

Gulrajani and Lopez-Paz [20] did extensive experiments on comparing DG algorithms, using the same neural architecture and data split. Specifically, they show that with data augmentation, ERM perform relatively well among a large array of algorithms. Our experiments are based on their settings. We run Algorithm 1 on standard benchmarks, including RotatedMNIST [18], PACS [28], Office-Home [51] and WILDS-FMoW [23] (c.f. Appendix C.1). Specifically, WILDS-FMoW is a large dataset with nearly half a million images. Detailed experimental settings can be seen at Appendix C.

Evaluating transferability From Figure 3 it can be seen that at a neighborhood of the learned classifier, there exists a classifier such that the target accuracy is degraded significantly, whereas some source domain still has high accuracy. This poses questions to whether current popular algorithms such as ERM [50], DANN [17] and Mixup [53, 54] are really learning invariant and transferable features. If so, the target accuracy should be high given a high source accuracy. However, for the PACS dataset and Mixup model (the second column of Figure 3), the target accuracy decreases by

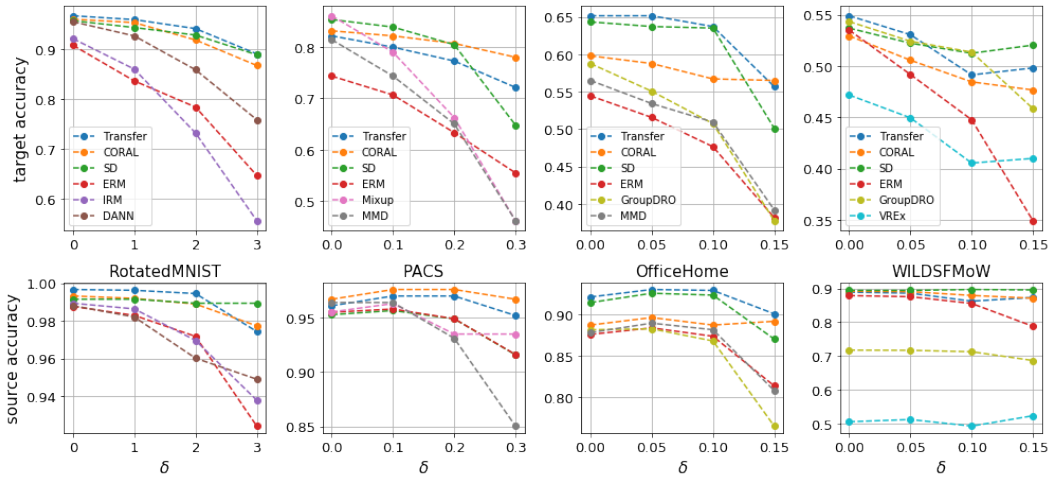

Figure 3: **Top row**: test accuracy of the target domain; **bottom row**: test accuracy of one of the source domains. Each column is for a given dataset with the name in the middle, and the legends on the bottom row are the same as those on the top row. δ is the parameter in Algorithm 1.

more than 30% while the source accuracy remains roughly at the same level. We can also, e.g., read from the first column that with a small decrease of the source (test) accuracy by $\sim 2\%$ (at $\delta = 2$), the target accuracy of DANN drops by $\sim 10\%$.

From Figure 3 we can also see that Correlation Alignment [CORAL, 47] and Spectral Decomposition [SD, 40] have better transferability that other algorithms. In some sense, they are in fact learning *robust classifiers*, i.e., all the classifiers on the neighborhood of the learned classifier can achieve good accuracies. With this robust classifier, the target accuracy does not decrease much even if the classifier is perturbed.

Improving transferability Algorithm 2 has good performance among all four datasets that we tried, comparable to CORAL and SD. Note that CORAL and SD do not always perform well, such as in the Office-Home and WILDS-FMoW datasets, but our Transfer algorithm does. However, in our experiments we find there are two limitations of Algorithm 2: **(1)** we need a large number of inner maximization steps to compute the gap, which needs more training time. This is similar to adversarial robustness [33] which is slower than usual training. In order to overcome this difficulty we used pretraining from other algorithms in the experiments on Office-Home and WILDS-FMoW; **(2)** Moderate hyper-parameter tuning is needed. For example, we need to tune N is Algorithm 2, the learning rate (`lr`) of SGA and the choice of δ. We find that taking $N = 20$ or 30 is usually a good choice, and δ can be quite large such that the projection step is not taken. We take $\texttt{lr} = 0.01$ for RotatedMNIST and $\texttt{lr} = 0.001$ for other datasets.

Label shift In order to show the difference with the well-known $\mathcal{H}$-divergence [8], we compute the label shifts in the PACS dataset. As shown in Zhao et al. [59], the optimal joint error $\epsilon_{\mathcal{S}}(h^*) + \epsilon_{\mathcal{S}}(h^*)$ (h^* is the optimal classifier that minimizes $\epsilon_{\mathcal{S}}(h) + \epsilon_{\mathcal{T}}(h)$) can be large under the shift of label distributions. We follow [59] and compute the label shift between pairs of domains in the PACS dataset, measured by total variation. From Table 1 we can see that the label shift is large in this case, and thus the $\mathcal{H}$-divergence bound [8] can be quite loose. Comparably, our transfer measure bound Prop. 3 is tighter (c.f. Prop. 27) and therefore still useful in practice.

Table 1: Label shift between pairs of domains in the PACS dataset. **TV:** total variation; **A:** art painting; **C**: cartoon; **P**: photo; **S**: sketch. The total variation is always between zero and one.

TV	**A**	**C**	**P**	**S**
A	0.0	0.12	0.11	0.3
C	0.12	0.0	0.18	0.24
P	0.11	0.18	0.0	0.37
S	0.3	0.24	0.37	0.0

6 Related Work

Multi-task learning Multi-task learning (MTL) [56] is related to but different from DG. In MTL, there are several tasks, and one hopes to improve the performance of each task by jointly training all the tasks simultaneously, utilizing the relationships between them. This is different from DG in the sense that in DG the target domain is unknown a priori, whereas in MTL the focus is more on better generalization on existing tasks that appear in training. Hence, there is no distribution shift in MTL per se. Furthermore, for MTL, the output spaces of different tasks are not necessarily the same.

Zero-shot learning / Few-shot learning / Meta-learning DG is different from zero-shot learning [27]. In zero-shot learning, one has labeled training data and the goal is to make predictions on a new unseen label set. However, in DG the label set remains the same for the source and the target domains. On the other hand, the focus of few-shot learning is on fast adaptation, in the sense that the test distribution remains the same as the training distribution, but the learner can only have access to a few labeled samples. Domain generalization also shares similarity with meta-learning. However, in meta-learning, the learner is allowed to fine-tune over the target domain. In other words, the protocol of meta-learning allows access to a small amount of labeled data from future unseen domains. Meta-learning is more or less one specific method that is used to tackle few-shot learning. Because of the similarity, some meta-learning algorithms can be applied to DG [29].

Self-supervised learning Self-supervised learning (SSL) is a popular unsupervised feature learning approach [13, 19, 21]. The goal of SSL is to learn invariant representations w.r.t. different views of the same image. Although it is a promising feature learning method, it differs from our DG settings in the sense that no labels are used in SSL.

Domain generalization There have been a lot of old and new algorithms proposed for domain generalization. The simplest one is Empirical Risk Minimization (ERM), where we simply minimize the empirical risk of (the sum of) all source domains. In Blanchard et al. [10], Muandet et al. [35], kernel methods for DG were proposed. Arjovsky et al. [3] proposed Invariant Risk Minimization (IRM) which aims to learn invariant predictors across source domains, and follow-up discussions can be found in [25, 41]. Another approach is called distributional robustness [43, 52], where the model is optimized over a worst-case distribution under the constraint that this distribution is generated from a small perturbation around the source distributions. In Albuquerque et al. [2], a DG scheme based on distribution matching was proposed. Moreover, many domain adaptation algorithms can be directly adapted to the task of domain generalization, such as CORAL [47] and DANN [16]. Last but not least, we mention a concurrent work [55] on the theory of domain generalization, which focuses more on proposing a model selection rule based on accuracy and variation.

Adversarial robustness Our evaluation and training methods in §4 are reminiscent of the adversarial training method [33] in the literature of adversarial robustness. Perturbing the classifier in our case corresponds to perturbing the input data in adversarial robustness. From this perspective, our Transfer algorithm is parallel to the adversarial training method. It would be interesting to design certified robust feature embeddings, by analogy with certified robust classifiers [15].

7 Conclusions

In this paper we formally define the notion of *transferability* that we can quantify, estimate and compute. Our transfer measures can be understood as a special class of IPMs. They are weaker than total variation and even very dissimilar distributions could be transferable to each other. Our definition of transferability can also be naturally used to derive a generalization bound for prediction error on the target domain. Based on our theory, we propose algorithms to evaluate and improve the transferability by learning feature representations. Experiments show that, somewhat surprisingly, many existing algorithms are not quite learning transferable features. From this perspective, our transfer measures offer a novel way to evaluate the features learned from different DG algorithms. We hope that our proposal of transferability could draw the community's attention to further investigate and better understand the fundamental quantity that allows robust models under distribution shifts.

Broader Impact Reliable domain generalization models are important for practice use. Our work points out the reliability issue of DG algorithms. It is worth mentioning that our evaluation method can only disprove the transferability and survival of our attack method should not be treated as a warranty. Misunderstanding of it could lead to potential harm in practical applications.

Acknowledgements and Funding Transparency Statement

We thank the anonymous reviewers for their constructive comments as well as the area chair and the senior area chair for overseeing the review process. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. We thank NSERC and the Canada CIFAR AI Chairs program for funding support. GZ is also supported by David R. Cheriton scholarship and research grant from Vector Institute. Finally, we thank Vector Institute for providing the GPU cluster.

References

[1] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. In *International Conference on Machine Learning*, pages 145–155. PMLR, 2020.

[2] Isabela Albuquerque, João Monteiro, Mohammad Darvishi, Tiago H Falk, and Ioannis Mitliagkas. Generalizing to unseen domains via distribution matching. *arXiv preprint arXiv:1911.00804*, 2019.

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

[4] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning*, pages 224–232. PMLR, 2017.

[5] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[6] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[7] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *J. Mach. Learn. Res.*, 20(63):1–17, 2019.

[8] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[9] 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.

[10] 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.

[11] Gilles Blanchard, Aniket Anand Deshmukh, Urun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *Journal of Machine Learning Research*, 22(2):1–55, 2021.

[12] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *Advances in neural information processing systems*, 2007.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607, 2020. URL `http://proceedings.mlr.press/v119/chen20j.html`.

[14] 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*, pages 6172–6180, 2018.

[15] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

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

[17] 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.

[18] 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.

[19] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html`.

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

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[22] Zeyi Huang, Haohan Wang, Eric P. Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *ECCV*, 2020.

[23] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. WILDS: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*, 2020.

[24] Vladimir Koltchinskii. Rademacher complexities and bounding the excess risk in active learning. *The Journal of Machine Learning Research*, 11:2457–2485, 2010.

[25] Masanori Koyama and Shoichiro Yamaguchi. When is invariance useful in an out-of-distribution generalization problem?, 2021.

[26] 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). *arXiv preprint arXiv:2003.00688*, 2020.

[27] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE, 2009.

[28] 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.

[29] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[30] 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*, pages 5400–5409, 2018.

[31] 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.

[32] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[34] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

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

[36] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, pages 429–443, 1997.

[37] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap via style-agnostic networks. *arXiv preprint arXiv:1910.11645*, 2019.

[38] Balas K Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

[39] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pages 947–1012, 2016.

[40] 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.

[41] Elan Rosenfeld, Pradeep Kumar Ravikumar, and Andrej Risteski. The risks of invariant risk minimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=BbNIbVPJ-42`.

[42] Walter Rudin. *Real and complex analysis*. McGraw-Hill Education, 1987.

[43] Shiori Sagawa*, Pang Wei Koh*, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=ryxGuJrFvS`.

[44] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[45] Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

[46] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, Gert RG Lanckriet, et al. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2012.

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

[48] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *Advances in Neural Information Processing Systems*, 33, 2020.

[49] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[50] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

[51] 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.

[52] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *NeurIPS*, 2018.

[53] 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.

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

[55] Haotian Ye, Chuanlong Xie, Tianle Cai, Ruichen Li, Zhenguo Li, and Liwei Wang. Towards a theoretical framework of out-of-distribution generalization, 2021.

[56] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021. doi: 10.1109/TKDE.2021.3070203.

[57] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413. PMLR, 2019.

[58] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. *Advances in neural information processing systems*, 31:8559–8570, 2018.

[59] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the Conclusion section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See AppendixA.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix C.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] We did try different random sends, but we chose the worst case scenario.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The license issue is discussed in Appendix C.1.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We are not aware of any personally identifiable information or offensive content in the datasets we use. See Appendix C.1.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]