Domain Generalization with Interpolation Robustness

Ragja Palakkadavath R.PALAKKADAVATH@DEAKIN.EDU.AU
Applied Artificial Intelligence Institute, Deakin University
Thanh Nguyen-Tang NGUYENT@CS.JHU.EDU
Whiting School of Engineering, Johns Hopkins University
Hung Le THAI.LE@DEAKIN.EDU.AU
Applied Artificial Intelligence Institute, Deakin University
Svetha Venkatesh SVETHA.VENKATESH@DEAKIN.EDU.AU
Applied Artificial Intelligence Institute, Deakin University
Sunil Gupta SUNIL.GUPTA@DEAKIN.EDU.AU
Applied Artificial Intelligence Institute, Deakin University

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Domain generalization (DG) uses multiple source (training) domains to learn a model that generalizes well to unseen domains. Existing approaches to DG need more scrutiny over (i) the ability to imagine data beyond the source domains and (ii) the ability to cope with the scarcity of training data. To address these shortcomings, we propose a novel framework - *interpolation robustness*, where we view each training domain as a point on a domain manifold and learn class-specific representations that are domain invariant across all interpolations between domains. We use this representation to propose a generic domain generalization approach that can be seamlessly combined with many state-of-the-art methods in DG. Through extensive experiments, we show that our approach can enhance the performance of several methods in the conventional and the limited training data setting.

Keywords: domain generalization ; limited data ; robustness ; latent interpolation ; invariant representation

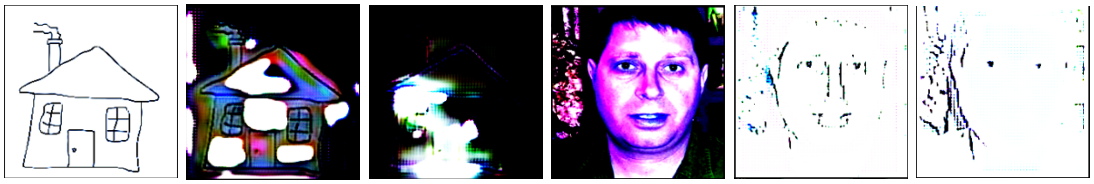
1. Introduction

Domain generalization (DG) (Muandet et al., 2013) aims at learning to generalize well to an unseen distribution given data from multiple source distributions. Training data is available from a set of related sources, where each source pertains to a set of labeled data from a specific domain distribution. Domain distributions can differ from each other in multiple ways. For example, the shift can occur in covariates, label generation mechanisms, or both. In this paper, we are concerned with generalization under covariate shift where domains differ in their covariate distributions.

Fig 1 displays images from PACS dataset (Li et al., 2017a), a benchmark dataset for domain generalization. Images belong to the classes **elephant** and **dog** and extend across four domains: *photo, art painting, cartoon, and sketch.* A good generalization algorithm trained to distinguish between these classes on the domains comprising *photos, cartoons, and sketches* should be able to successfully classify the data in a new domain, e.g., *art painting.*

(a) *photo* (b) *art painting* (c) *cartoon* (d) *sketch*

Figure 1: Sample images from the PACS dataset - the images are from the **dog** and **elephant** class and belong to the domains: *photo, art painting, cartoon, and sketch.*



(a) original (sketch) (b) translated (photo) (c) translated (limited data) (d) original (photo) (e) translated (sketch) (f) translated (limited data)

Figure 2: An illustration of limited data problem in DG. The power of domain density transformations in DIRT reduces when training data is limited. The translated images from the StarGAN model of DIRT become too incoherent for proper identification when the training data (PACS dataset) is scaled down. Hence, domain density transformation alone is insufficient for a well-generalized classifier in the limited data setting.

An effective way to generalize from multiple sources is to find common patterns across all the source domains. Many of the popular models in DG such as DIRT (Nguyen et al., 2021), MBDG (Robey et al., 2021), DDAIG (Zhou et al., 2020) do so by learning transformations between source domains by mapping the data from one domain to another and enforce their underlying representations to be similar and, therefore, domain invariant. The transformations are learned using generative models (e.g., StarGAN (Choi et al., 2018), CycleGAN (Zhu et al., 2020), MUNIT (Huang et al., 2018)).

Imagining beyond source domains. One of the limitations of domain invariant transformation models is their inability to imagine data beyond the source domains. Since the goal of DG is to perform well in a target domain, performing translations between source domains alone may *overfit* the model's imagination capability solely to the training data. As an example, DIRT learns the invariant representation by enforcing the representations of an instance and its domain-translated versions to be similar. If it were to transform an image of a dog in the photo domain to the sketch domain, it aligns the photo and sketch closer but does not know about other domains around it.

Failure of existing models under limited data. Domain invariant transformation models translate data from one domain to another using generative models such as StarGAN (Choi et al., 2018), CycleGAN (Zhu et al., 2020), MUNIT (Huang et al., 2018) that require a substantial amount of training data. However, there is a shortage of labeled training data in many critical applications (e.g., medical image analysis (Castro et al., 2020)). Solutions for DG must address this challenge while also being robust with unseen

data. Existing solutions often assume sufficient training data, but training is difficult when supervised data is scarce. As illustrated in Fig. 2, the domain translations performed by StarGAN (used in DIRT (Nguyen et al., 2021)) on images from the PACS dataset are not very effective when the training data reduces to 5% ($\sim$ 500 images). This problem can trickle down to the generalization capabilities of the DG model in the limited data setting.

To overcome the above limitations, we propose a novel approach called *Interpolation Robustness* which addresses DG in the conventional and the limited data setting. Our key perspective is that we view a domain distribution as an interpolation between the distributions of other domains, and by learning a representation that is invariant and robust under such interpolation, we can equip the model to have better generalization in unseen domains. To realize this insight in practice, we interpolate between *latent* representations of a pair of inputs from two different domains that have the same class label by varying the mixing coefficient. We assign the same class label to every intermediate representation on the interpolation path. This approach allows the model to learn a representation that is invariant for the source domains and invariant across all the interpolated domains. It leads to better domain generalization, especially when the unseen domain falls around the interpolated domains. This can be conceptualized as expanding from the source domains to the full span of their interpolations.

More abstractly, if the domain distributions were points on a manifold, we can recover many points not initially present in the training data through interpolation. Our concept of interpolation robustness can be *valuable in scenarios with limited training data*. When there is insufficient training data, we need a method that ensures robust representations of images in the latent space across various imaginary domains. Interpolation achieves this by *densifying* the distributions of existing domains via creating new domains through interpolation. Our main contributions are:

- We propose *interpolation robustness*, a novel framework for DG that aims at learning a representation that is robust against interpolation between different domains;
- We create a practical model-agnostic approach that one can use along with any of the existing DG algorithms that use domain-invariant representation;
- We show that our method can significantly enhance the performance of state-of-the-art domain invariant representation-based models on PACS, VLCS, RotatedMNIST, and OfficeHome datasets for the DG task in *conventional and limited data* settings.

2. Related Works

2.1. Invariant Representation Learning

One of the popular approaches to DG is to learn a domain-invariant representation (Arjovsky et al., 2019; Li et al., 2018; Nguyen et al., 2021; Zhao et al., 2020) from the training domains. This stems from the assumption that the differences across domains are irrelevant to the classification task and there exists a common representation that captures all the domains. The aim is to extract a domain-agnostic model from these domains which generalizes to an unseen domain. Existing DG algorithms transform data into a representation that captures the essence shared by all the domains while eliminating non-generalizable features from

individual domains. Minimizing the domain divergences between any training domain and its domain-translated version has been a recent area of research. It has achieved effective performance (Nguyen et al., 2021; Robey et al., 2021; Zhou et al., 2020) in DG with covariate shift. Large generative models (e.g., GANs) are used to perform domain translation. Our studies show that training a generative model is less effective when the training data is limited. Since the domain translations are only among the source domains, there is a good chance of these hugely parameterized models overfitting to the training data.

2.2. Training Strategy

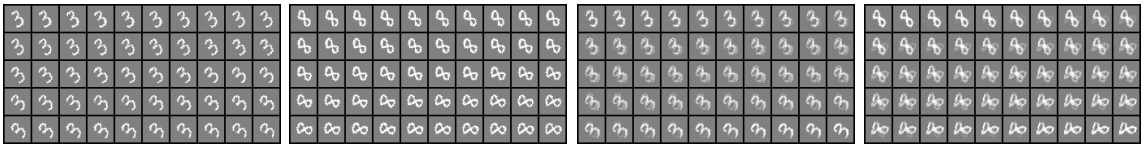
Prior works have used different training strategies to promote generalization capabilities. Examples of such models are meta learning (Li et al., 2017b; Balaji et al., 2018) and ensemble learning (Wang et al., 2020a; Mancini et al., 2018; Cha et al., 2021). Meta-learning in DG divides the source domains into multiple train and test sets. The models are trained on the train set and made to adapt to the test set. Since they have been trained to adapt to unseen data, they are also expected to generalize well to an actual unseen domain. In ensemble learning, multiple domain-specific models are learned for each source domain. Then these models are combined in different heuristic ways to make predictions for the unseen domain to improve the generalization ability. Other than assembling model predictions, weights of the model obtained across different epochs are averaged for better generalization.

2.3. Data Augmentation Techniques

Augmenting the training data with additional information can enhance the diversity of training data. Heterogeneity brought by data augmentation could prevent overfitting and improve the generalization ability. Additional information can be created through adversarial data augmentation (Zhou et al., 2020; Volpi et al., 2018), through style transfers (Li et al., 2020; Gong et al., 2019), using optimal transport techniques (Zhou et al., 2021).

Recently, few works (Yao et al., 2022; Wang et al., 2020b; Yan et al., 2020) have adapted Mixup (Zhang et al., 2018) to address the problem of subpopulation shift and distribution shift. These methods mix the input and their labels in the pixel space. However, images created through the combination of other images in the pixel space can cause a complex arrangement of latent representations and lead to non-smooth classifier boundaries (Verma et al., 2018). They may contain noisy artifacts caused by the mixing of two different images (e.g., see Fig. 3(b)). Moreover, augmenting high-dimensional interpolated images with the training data puts an additional burden on the training procedure. Manifold Mixup (Verma et al., 2018) combines data in a latent space instead of pixel space. It has shown better performance over pixel space mixup for *in-distribution classification setting*. However, Manifold Mixup has not been used in a DG context.

In contrast to Yao et al. (2022); Wang et al. (2020b); Yan et al. (2020) that address distribution shift by mixing samples in the pixel space, *we perform the interpolation in a latent space*. Fig. 3 shows that interpolation in a latent space leads to better-quality images. Despite combining representations of images in the latent space, our work differs from Manifold Mixup as follows. (i) Instead of combining the input representation and their label in the latent space for a fixed coefficient, *we create a span of interpolated domains between the source domains, which is more informative to the model.* (ii) Instead of interpolating between



(a) Latent Space Interpolation (b) Pixel Space Interpolation

Figure 3: Fig. 3(a) shows the interpolated versions of a digit (class) between two angles (domain) in the latent space. For example, we interpolate digit 3's angle (domain) from 15° to 30° at a step of 0.02. Similarly, we create interpolations of digit 8 between 30° to 45° at the step of 0.02. Fig. 3(b) shows the same in pixel space. As seen in Fig. 3, the interpolation results in images that lie between 15° to 30° for digit 3 and 30° to 45° for digit 8. These angles (domains) were not present in the training data. Images obtained show promise of discovery of new domains originally not present in the data from interpolation.

pair of random representations, *we only interpolate between pairs of latent representations from the same class and different domains.* Our class-consistent interpolation establishes an invariant property that clusters representations of the same class from different domains.

3. Domain Generalization via Interpolation Robustness

Problem setting. We consider the standard setting of DG where data from multiple labeled source domains $\mathcal{S} = \{\mathcal{S}_1 \cup \mathcal{S}_2 \ldots \cup \mathcal{S}_m\}$ where $\mathcal{S}_d \sim \mathcal{P}_d$ is a collection of domain d (where $d \in \mathcal{D}$) data of the form $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathcal{X}$ is the input, $y \in \mathcal{Y}$ is the label of $\mathbf{x}$. The goal of a DG learner is to learn from $\mathcal{S}$ and generalize well to a novel domain $\tilde{d}$ that was unseen during training.

Motivation for Interpolation Robustness. Domain invariant transformation models lack the capability to envision data beyond the source domains, which may hinder their effectiveness in achieving DG.

We illustrate our interpolation robustness idea through an example. We use the RotatedMNIST dataset, where the domains are the angles of rotation. It contains 6 domains as follows: [0°, 15°, 30°, 45°, 60°, 75°]. The task is to perform well in an unseen domain. As seen from Fig. 3(a), an encoded representation of digit 3 at 15° and at 30° are interpolated at intervals of 0.02 to generate the representations between 15° and 30°. Due to class-consistent, robust interpolation, the intermediate image representations continue to be digit 3. However, they are at different rotation angles between 15° and 30°, which were not present in the training data. Intermediate representation may not always result in a domain close to the target domain, but it provides the model with rich information different from the training data. Such representations can enable the model to imagine different areas of the latent space for which no source data is present and fill up its knowledge of the latent space of possible domain distributions[1].

1. At a high level, Nguyen et al. (2023) also share our idea of capturing all possible domain distributions, though for a different problem of test-time adaptation. In addition, their imagined domain distributions are constructed from a set of transformations on the feature space (using adversarial samples), not in the latent space using interpolation as in our framework.

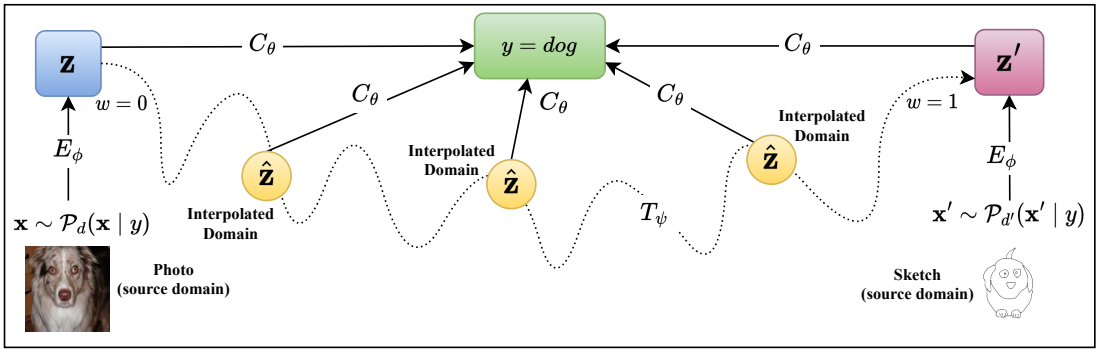


Figure 4: A schematic of interpolation robustness. Image instances $\mathbf{x}$ and $\mathbf{x}'$ are sampled from the same class y but with different domain identifiers ($d \neq d'$). They are passed through encoder (E_ϕ) to get latent representations $\mathbf{z}$ and $\mathbf{z}'$. Intermediate samples $\hat{\mathbf{z}}$ are obtained by interpolating from $\mathbf{z}$ to $\mathbf{z}'$ by varying w from 0 to 1. Classifier (C_θ) takes $\mathbf{z}$, $\mathbf{z}'$ and $\hat{\mathbf{z}}$ to the same class y to ensure robustness against interpolation operation.

3.1. Design

We describe the practical realization of interpolation robustness below. It has 3 components.

Encoder: We consider an encoder as $E_\phi : \mathcal{X} \to \mathcal{Z}$, parameterized by ϕ. It maps data from the high-dimensional pixel space into a low-dimensional latent space. E_ϕ must create the latent representation space $\mathcal{Z}$ such that the representations $\mathbf{z}$ obtained from inputs that share the same label should lie clustered together and $\mathbf{z}$'s obtained from inputs having different labels should lie spread apart. The classifier module below ensures that property.

Classifier: We consider a classifier as $C_\theta : \mathcal{Z} \to \mathcal{Y}$, parameterized by θ. $C_\theta(\mathbf{z})$ predicts the class of samples $\mathbf{x}$ from their latent representation $\mathbf{z}$ via the encoder E_ϕ. For C_θ to associate the representation $\mathbf{z}$ to its correct label, we minimize the following loss function:

$$\mathcal{L}_{\text{cls}}(\theta, \phi) = \mathbb{E}\left[l(C_\theta(E_\phi(\mathbf{x})), y)\right], \tag{1}$$

where $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the training loss (e.g., cross-entropy), and the expectation is taken with respect to $\mathbf{x} \sim \mathcal{P}_d(\mathbf{x} \mid y), y \sim \mathcal{P}_d(y), d \sim \mathcal{D}$.

Interpolator: Instances $\mathbf{x}$ and $\mathbf{x}'$ with the same label y are sampled from domains d and d' respectively, i.e. $\mathbf{x} \sim \mathcal{P}_d(\mathbf{x} \mid y)$ and $\mathbf{x}' \sim \mathcal{P}_{d'}(\mathbf{x}' \mid y)$. The encoded latent representations are obtained as $\mathbf{z} = E_\phi(\mathbf{x})$ and $\mathbf{z}' = E_\phi(\mathbf{x}')$. Inputs to the interpolator are $\mathbf{z}$, $\mathbf{z}'$ and interpolation weight w. The interpolator outputs a latent representation $\hat{\mathbf{z}} = I(\mathbf{z}, \mathbf{z}', w)$. The set of interpolated samples $\{\hat{\mathbf{z}}\}$ that define the interpolation path is obtained by sampling w at uniform intervals from 0 through 1. Interpolation can be performed in multiple ways. Eq. 2 below is an example of linear interpolation.

$$\hat{\mathbf{z}} = I(\mathbf{z}, \mathbf{z}', w) = \mathbf{z} + w\,(\mathbf{z}' - \mathbf{z}) \tag{2}$$

We have posited that the domains lie on a low-dimensional manifold. Using a strictly linear interpolation on this manifold may be rigid and only allow the discovery of domains on linear

Algorithm 1 *DNT: **D**eepAll with **INT**erpolation Robustness*

Require: Training data: $\mathcal{S}$, batch size: B, learning rate: η, hyperparameter λ

Initialize the weights of the neural networks: θ, ϕ and ψ
for epoch in *MAX_EPOCH* **do**
  Sample batches $\{\mathbf{x}_i, y_i, d_i\}_{i=1}^{B}$, $\{\mathbf{x}'_i, y'_i, d'_i\}_{i=1}^{B}$ from $\mathcal{S}$ such that $d'_i \neq d_i$ and $y_i = y'_i$
  for each i **do**
    $\mathbf{z}_i \leftarrow E_\phi(\mathbf{x}_i)$ *# Compute the representations via the encoder*
    $\mathbf{z}'_i \leftarrow E_\phi(\mathbf{x}'_i)$
    $\{\hat{\mathbf{z}}_i\} \leftarrow \emptyset$, $\{y_i\} \leftarrow \emptyset$
    for $w \sim \text{Unif}([0, 1])$ **do**
      $\hat{\mathbf{z}}_i \leftarrow \mathbf{z}_i + w\, T_\psi(\mathbf{z}'_i - \mathbf{z}_i)$ *# Compute the interpolations*
      $\{\hat{\mathbf{z}}_i\} \leftarrow \{\hat{\mathbf{z}}_i\} \cup \hat{\mathbf{z}}_i$ *# Stack representations for the interpolation path*
      $\{y_i\} \leftarrow \{y_i\} \cup y_i$
    end for
  end for
  $\mathcal{L}_{\text{cls}}(\theta, \phi) \leftarrow \sum_i [l(C_\theta(\mathbf{z}_i), y_i)]$ *# Compute the classification loss*
  $\mathcal{L}_{\text{int}}(\theta, \phi, \psi) \leftarrow \sum_i [l(C_\theta(\{\hat{\mathbf{z}}_i\}), \{y_i\}) + \|T_\psi(\mathbf{z}'_i - \mathbf{z}_i) - (\mathbf{z}'_i - \mathbf{z}_i)\|_2]$
  $\mathcal{L}_{\text{dnt}} \leftarrow \mathcal{L}_{\text{cls}}(\theta, \phi) + \lambda\, \mathcal{L}_{\text{int}}(\theta, \phi, \psi)$
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{dnt}}$
  $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}_{\text{dnt}}$
  $\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}_{\text{dnt}}$ *# Update the parameters via gradient descent*
end for
return Trained θ, ϕ and ψ

paths. This might cause many domains to be left unexplored. To add flexibility, we extend from linear to nonlinear interpolation using a parametric function T_ψ such that $T_\psi : \mathcal{Z} \to \mathcal{Z}$. Given this, we have our definition for nonlinear interpolation below:

$$\hat{\mathbf{z}} = I(\mathbf{z}, \mathbf{z}', w, T_\psi) = \mathbf{z} + w\, T_\psi(\mathbf{z}' - \mathbf{z}) \tag{3}$$

T_ψ for linear interpolation is an identity mapping. In the nonlinear case, a neural network can be used model T_ψ. However, at $w = 1$, $\hat{\mathbf{z}}$ should be close to $\mathbf{z}'$ ensuring that the terminal of the interpolation curve is $\mathbf{z}'$. This is realized by minimizing: $\|T_\psi(\mathbf{z}' - \mathbf{z}) - (\mathbf{z}' - \mathbf{z})\|_2$.

Given $\mathbf{z}$ and $\mathbf{z}'$, we enforce the interpolated samples $\hat{\mathbf{z}}$ to have the same label under interpolation, i.e. $\hat{\mathbf{z}}$ should yield the same label y as its reference representations $\mathbf{z}$ and $\mathbf{z}'$. We define the interpolation robustness loss (INT) below to encode such behavior.

$$\mathcal{L}_{\text{int}}(\theta, \phi) = \mathbb{E}\left[l\left(C_\theta(I(\mathbf{z}, \mathbf{z}', w, T_\psi)), y\right) + \|T_\psi(\mathbf{z}' - \mathbf{z}) - (\mathbf{z}' - \mathbf{z})\|_2\right], \tag{4}$$

where the expectation is taken with respect to $y \sim \mathcal{P}_d(y), (d, d') \overset{i.i.d.}{\sim} \mathcal{D}, \mathbf{x} \sim \mathcal{P}_d(\mathbf{x} \mid y), \mathbf{x}' \sim \mathcal{P}_{d'}(\mathbf{x}' \mid y)$ and $w \in \text{Unif}([0, 1])$. Thus, to train E_ϕ, C_θ and T_ψ we jointly minimize the loss in Eq. (5). Our final loss function is:

$$\mathcal{L}_{dnt}(\theta, \phi, \psi) = \mathcal{L}_{\text{cls}}(\theta, \phi) + \lambda\, \mathcal{L}_{\text{int}}(\theta, \phi, \psi), \tag{5}$$

where λ is a regularization hyperparameter. Training a DG model with only $\mathcal{L}_{\text{cls}}$ is the

Table 1: Baselines and Our models

Baseline	Our corresponding model variant
DeepAll : $\mathcal{L}_{\text{cls}}$	DNT: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{int}}$
Mixup: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{mixup}}$	DNT: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{int}}$
Manifold Mixup: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{manifoldmixup}}$	DNT: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{int}}$
DIRT: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{dirt}}$	DRINT: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{dirt}}$, $\mathcal{L}_{\text{int}}$
DGER: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{dger}}$	DGNT: $\mathcal{L}_{\text{cls}}$, $\mathcal{L}_{\text{dger}}$, $\mathcal{L}_{\text{int}}$

same as performing empirical risk minimization on the aggregated training data across all domains. It is referred to in the literature as DeepAll (Zhang et al., 2022; Nguyen et al., 2021; Carlucci et al., 2019; Dou et al., 2019). It is used as a building block for every DG algorithm. *We refer to the standalone variant of our interpolation robustness model with nonlinear interpolation as* ***DNT*** *(DeepAll + iNTerpolation :* $\mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{int}}$*) and describe it in Algorithm 1.* Our method can also be used as a **meta approach** with other DG methods that use invariant representation. Namely, we consider DIRT (Nguyen et al., 2021), DGER (Zhao et al., 2020). We describe these variants as DIRT with interpolation robustness; **DRINT**: $(\mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{int}} + \mathcal{L}_{\text{dirt}})$ and DGER with interpolation robustness; **DGNT**: $(\mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{int}} + \mathcal{L}_{\text{dger}})$. $\mathcal{L}_{\text{dirt}}$ and $\mathcal{L}_{\text{dger}}$ were proposed in Nguyen et al. (2021) and Zhao et al. (2020) respectively.

3.2. Experiments

We perform experiments to demonstrate the effectiveness of our proposed method. We discuss baseline methods, datasets used, and the experimental settings. We also provide the experimental results, followed by a few visualizations of the latent space.

3.2.1. Baselines

DeepAll performs empirical risk minimization over the aggregated training data from all the domains. It is commonly used as a baseline in DG literature (Carlucci et al., 2019; Dou et al., 2019; Nguyen et al., 2021; Zhang et al., 2022).

DIRT (Nguyen et al., 2021) enforces invariance on $\mathbf{z}$ through domain invariant density transformations and is one of the best performing models in its model family. We chose this model as one of our baselines to demonstrate its performance deterioration due to limited training data.

DGER (Zhao et al., 2020) enforces conditional invariance ($p_i(y \mid \mathbf{z} = f(\mathbf{x})) = q(y \mid \mathbf{z} = f(\mathbf{x}))$ for all domains i on the representation $\mathbf{z}$ via an entropy regularization term. We chose DGER as a baseline because it combines multiple regularization techniques like entropy regularization and adversarial learning, and we want to show that our method can be used with a complex model that performs invariant representation learning to improve its performance.

DGER and DIRT perform invariant representation learning. We use our interpolation idea in combination with the representation learning used in these respective methods. We compare DeepAll, DIRT, and DGER with our model variants **DNT**, **DRINT**, and **DGNT**, respectively. We also compare our method with two Mixup based baselines described below.

Mixup-DG (Yan et al., 2020) combines images and their labels in the input space within samples in the training data to solve the problem of distribution shift. We included Mixup-DG (referred to as Mixup from now) as a baseline because it also utilizes interpolation to improve generalization in DG.

Manifold Mixup-DG (Verma et al., 2018) is a mixup-based model that performs interpolation in latent space instead of the input space. Manifold Mixup performs better than Mixup for in-distribution generalization. However, it has not been used in the context of DG. We *adapt* Yan et al. (2020) by replacing the pixel-based interpolation with latent space interpolation to create a Manifold Mixup baseline for DG (referred to as Manifold Mixup from now). We compare Mixup and Manifold Mixup with **DNT**, our model discussed in Algorithm 1. We summarize the baselines and our model variants in Table 1.

3.2.2. Datasets

We ran our experiments on 4 benchmark datasets in DG: PACS, VLCS, RotatedMNIST, and OfficeHome. We discuss 3 of them here. Experiments on the OfficeHome dataset are discussed in the supplementary material.

PACS (Li et al., 2017a) dataset consists of 9991 images from 7 classes. These images come from one of the four domains: art painting (A), cartoon (C), photo (P), and sketch (S). This dataset provides a good generalization gap between the domains.

VLCS (Ghifary et al., 2015) dataset contains $10,729$ images from 4 domains, where each domain itself is a dataset of natural images. They are VOC2007 (V), LabelMe (L), Caltech-101 (C), and SUN09 (S). The images are distributed across 5 classes.

RotatedMNIST (Ghifary et al., 2015; Ilse et al., 2020). Each domain of this dataset consists of 1000 samples of MNIST digits (LeCun and Cortes, 2010) with 100 from each class rotated at a specific angle. There are 6 such domains with the following degrees of rotation: $0°, 15°, 30°, 45°, 60°$ and $75°$.

3.2.3. Experimental Settings.

After splitting the dataset into train, validation, and test splits, we trained our models on the training data. To address the limited data scenario, we opted to reduce the size of our training dataset. We followed proportional sampling that does not modify the distribution of the class labels across domains ($P_d(y)$). For PACS, VLCS, and RotatedMNIST datasets, we randomly sampled subsets from each of the following sizes: 20%, 10%, and 5% of the training data while keeping $P_d(y)$ intact. 5% was the smallest subset we could choose while ensuring a representative sample from every class and domain. We scaled down training and validation data. We did not modify the test data.

Implementation Details. For the PACS dataset, we chose E_ϕ as Resnet-18 (He et al., 2016) for experiments related to DeepAll, DIRT, and DGER and Resnet-50 for Mixup and Manifold Mixup. For the VLCS dataset, we chose Alexnet (Krizhevsky et al., 2012) as E_ϕ for the experiments related to DIRT and DeepAll, Resnet-18 for DGER, and Resnet-50 for Mixup and Manifold Mixup. For the RotatedMNIST dataset, we used the standard MNIST CNN as E_ϕ across all models, where the feature network consists of two convolutional layers

Table 2: Prediction accuracy % on PACS. Our methods: DNT, DRINT, and DGNT outperform DeepAll, DIRT, and DGER, respectively

PACS						
Model	E_ϕ	**100%** Acc ± Std Err	**20%** Acc ± Std Err	**10%** Acc ± Std Err	**5%** Acc ± Std Err	**Average** Acc
DeepAll	Resnet 18	80.25 ± 0.5	69.17 ± 0.8	64.77 ± 1.3	58.50 ± 2.2	68.18
DNT		**82.92** ± 0.4	**72.62** ± 1.2	**68.66** ± 1.3	**61.29** ± 1.8	**71.37**
DIRT	Resnet 18	82.81 ± 0.3	73.02 ± 0.7	66.58 ± 1.6	59.98 ± 2.6	70.60
DRINT		**84.03** ± 0.4	**74.63** ± 1.0	**70.56** ± 1.3	**64.80** ± 2.0	**73.50**
DGER	Resnet 18	80.85 ± 0.4	73.80 ± 1.0	69.79 ± 1.2	65.00 ± 1.5	72.36
DGNT		**81.08** ± 0.4	**75.00** ± 1.0	**72.31** ± 1.0	**68.24** ± 1.5	**74.16**

Table 3: Prediction accuracy % on VLCS. Our methods: DNT, DRINT, and DGNT outperform DeepAll, DIRT, and DGER, respectively.

VLCS						
Model	E_ϕ	**100%** Acc ± Std Err	**20%** Acc ± Std Err	**10%** Acc ± Std Err	**5%** Acc ± Std Err	**Average** Acc
DeepAll	Alexnet	71.56 ± 0.8	68.71 ± 0.9	67.81 ± 1.3	63.13 ± 1.1	67.80
DNT		**72.93** ± 0.4	**69.88** ± 1.2	**68.68** ± 1.1	**63.60** ± 1.5	**68.77**
DIRT	Alexnet	73.11 ± 0.4	69.91 ± 1.0	68.19 ± 1.6	63.54 ± 1.3	68.69
DRINT		**73.57** ± 0.4	**70.21** ± 0.7	**68.46** ± 1.4	**65.93** ± 1.2	**69.54**
DGER	Resnet 18	75.87 ± 0.4	73.84 ± 0.5	72.72 ± 0.7	71.24 ± 1.0	73.42
DGNT		**76.47** ± 0.4	**74.03** ± 0.5	**72.77** ± 0.7	**71.72** ± 1.0	**73.74**

and one dense layer. C_θ is a single layer dense network preceded by a ReLU layer activation for all datasets across all models. *We made these modeling choices for a fair comparison with the baselines.* T_ψ is a 3 layer convolutional neural network. Experimental details like the dimension of the latent space, optimization algorithm used, hyperparameter λ, and other training information are provided in the supplementary material. Since we are the first to introduce a limited data setting in DG, we had to re-train all the baselines.

3.2.4. Experimental Results

Accuracy on test domains. We used the leave-one-domain-out evaluation scheme, where one of the domains is chosen as the target domain and kept aside for evaluation and the model is trained on the rest of the domains. We repeated this procedure for every domain in the dataset. Data from the training domains were randomly split into the train (90%) and validation (10%) sets. We computed test accuracy corresponding to the model with the best validation set performance. We repeated the experiments over 5 runs to calculate the mean accuracy and standard error. Finally, we report the average prediction accuracy obtained over all the domains on the baselines: DeepAll, DIRT, and DGER, and our respective counterparts: DNT, DRINT, and DGNT in Table 2, Table 3, and Table 4 for PACS, VLCS, and RotatedMNIST, respectively. Prediction accuracies on the individual domains are provided in the supplementary material.

Table 4: Prediction accuracy % on RotatedMNIST. Our methods: DNT, DRINT, and DGNT outperform DeepAll, DIRT, and DGER, respectively.

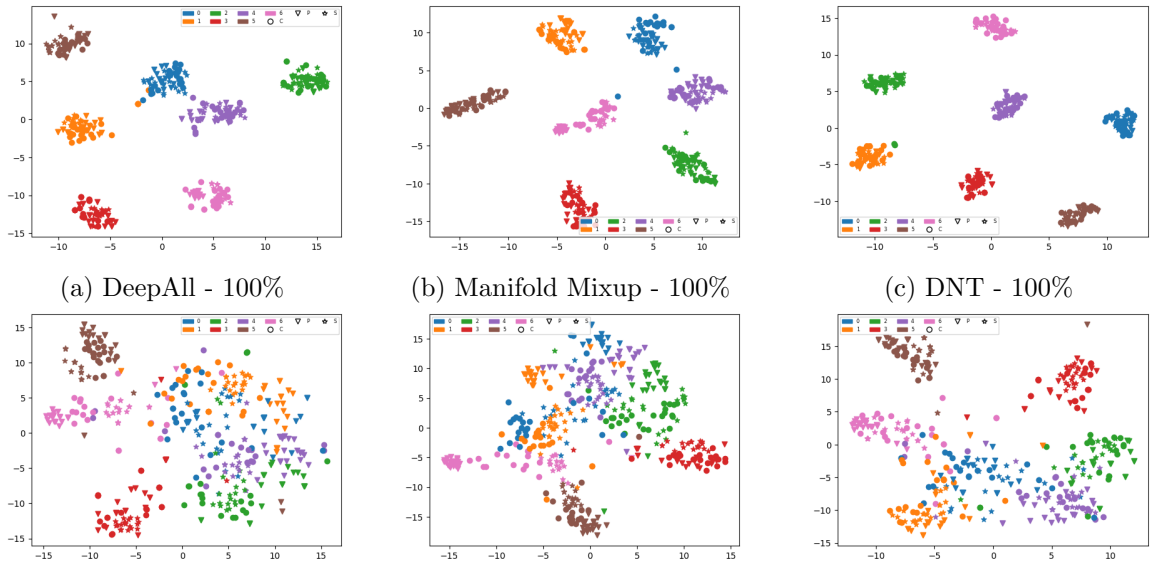
RotatedMNIST						
Model	E_ϕ	**100%** Acc ± Std Err	**20%** Acc ± Std Err	**10%** Acc ± Std Err	**5%** Acc	**Average**
DeepAll	MNIST CNN	92.69 ± 0.3	80.80 ± 0.6	73.95 ± 0.8	67.99 ± 0.8	78.86
DNT		**97.36** ± 0.1	**84.48** ± 0.6	**78.89** ± 0.6	**73.51** ± 0.5	**83.66**
DIRT	MNIST CNN	98.75 ± 0.1	84.95 ± 0.4	77.29 ± 0.8	70.24 ± 0.9	82.81
DRINT		**98.83** ± 0.1	**86.15** ± 0.3	**79.36** ± 0.5	**74.27** ± 0.6	**84.65**
DGER	MNIST CNN	95.61 ± 0.1	79.89 ± 0.5	73.69 ± 0.4	68.78 ± 0.8	79.49
DGNT		**95.92** ± 0.1	**83.85** ± 0.5	**77.27** ± 0.4	**72.38** ± 0.8	**82.36**

Table 5: Comparison of DNT with other interpolation-based methods

			100%	**20%**	**10%**	**5%**	**Average**
Dataset	Model	E_ϕ	Acc ± Std Err	Acc ± Std Err	Acc ± Std Err	Acc ± Std Err	Acc
PACS	Mixup	Resnet 50	84.60 ± 0.6	75.81 ± 1.6	71.84 ± 3.0	57.93 ± 2.8	72.54
	Manifold Mixup		84.20 ± 1.1	76.07 ± 1.5	73.40 ± 2.2	65.77 ± 3.7	74.86
	DNT		**86.57** ± 0.5	**77.42** ± 0.7	**73.71** ± 1.1	**66.92** ± 2.6	**76.15**
VLCS	Mixup	Resnet 50	76.84 ± 1.2	73.16 ± 1.6	72.34 ± 1.9	66.77 ± 2.6	72.28
	Manifold Mixup		**77.58** ± 1.0	74.16 ± 1.7	72.44 ± 1.2	66.70 ± 2.4	72.72
	DNT		77.56 ± 0.5	**76.08** ± 0.5	**75.13** ± 0.9	**73.66** ± 1.0	**75.61**
Rotated MNIST	Mixup	MNIST CNN	92.98 ± 0.3	80.81 ± 0.6	75.60 ± 0.4	69.80 ± 0.8	79.80
	Manifold Mixup		92.98 ± 0.3	80.81 ± 0.5	75.56 ± 0.4	69.85 ± 0.8	79.80
	DNT		**97.36** ± 0.1	**84.48** ± 0.6	**78.89** ± 0.6	**73.51** ± 0.5	**83.66**

Mean accuracy and standard error for the complete training data are under the column 100% of Tables 2, 3, and 4. We report the mean accuracy and standard error when the training dataset size is scaled down to 20%, 10%, and 5% of its original size. Moving rightwards over the columns under each split, we can observe an accuracy gain from just over 1% to 5% between DIRT and DRINT for the PACS dataset. A similar trend follows for VLCS and RotatedMNIST datasets with some variations. *Our DNT variant alone outperforms DIRT when the data is low, and it confirms our hypothesis that the performance of DIRT is worse than other methods with limited training data.* However, through interpolation robustness loss, we overcome this limitation in DRINT. The other baselines: DeepAll and DGER, also experienced a reduction in their performance with lesser data, and it can be seen from the tables that our models, through interpolation loss, their performance across all splits of data improved. We observed *lower standard errors* for our methods compared to the baselines, a characteristic of a stable model.

Comparison with Mixup based methods. We compared our model DNT with Mixup (Yan et al., 2020) and Manifold Mixup. Mixup was originally proposed for domain adaptation, and Gulrajani and Lopez-Paz (2021) adapted it for domain generalization. We followed their setup and used Resnet-50 to model our E_ϕ. We tuned the mixing coefficient hyperparameter across (0.1, 0.2, 0.5, 0.9) and picked 0.1. We report the mean accuracy and standard error for PACS, VLCS, and RotatedMNIST datasets in Table 5.



(a) DeepAll - 100% (b) Manifold Mixup - 100% (c) DNT - 100%

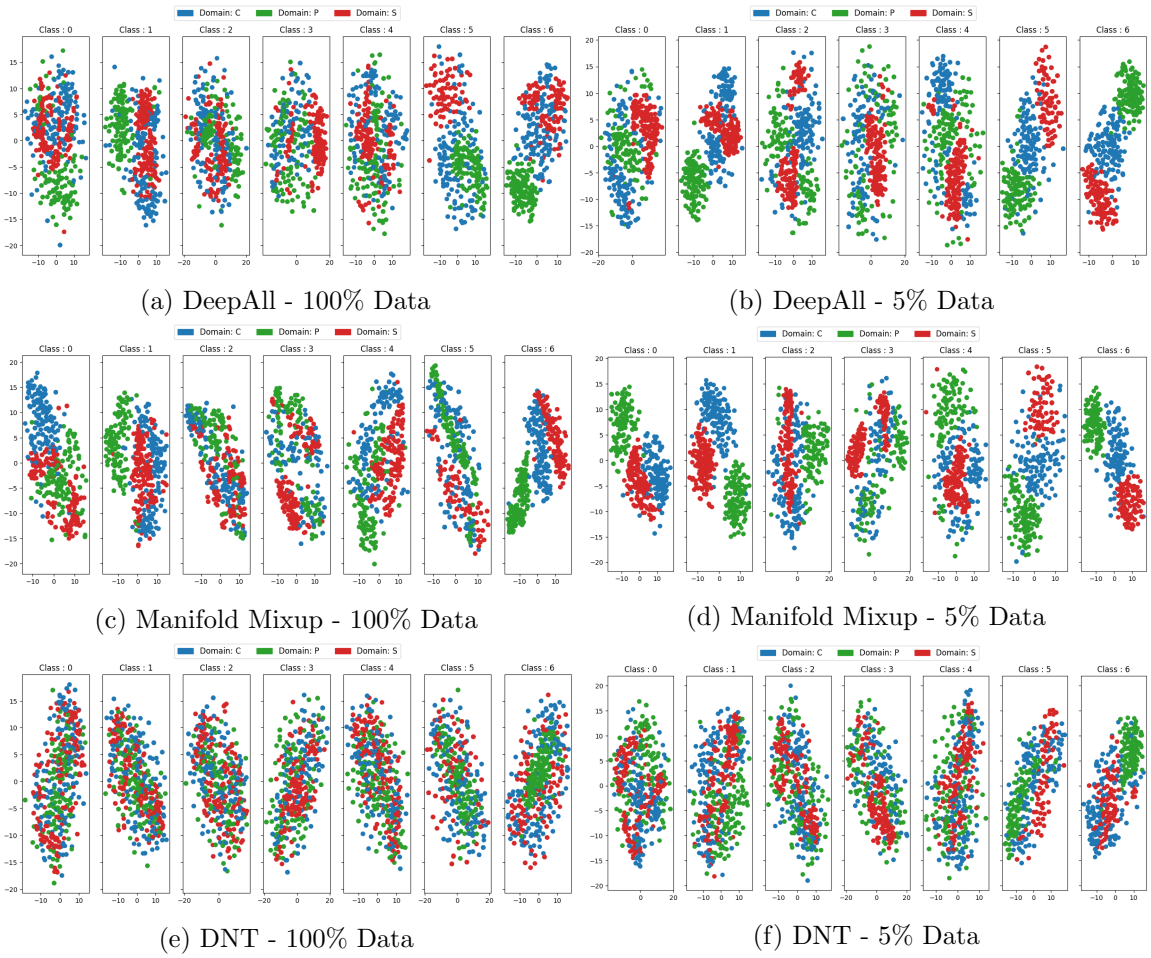
(d) DeepAll - 5% (e) Manifold Mixup - 5% (f) DNT - 5%

Figure 5: 2D visualization of $\mathbf{z}$. Fig. 5 (a), (b), and (c) show $\mathbf{z}$ where source domains were trained with 100% of data. Fig. 5 (d), (e), and (f) show $\mathbf{z}$ trained with 5% of data. Colors denote classes and markers denote domains. We can observe that the classes are better separated among the source domains in DNT over DeepAll and Manifold Mixup.

The results show that DNT fares better than Mixup and Manifold Mixup in the full data settings and the limited data settings. Unlike the baselines, we performed interpolation of data only within the same class. We targeted invariance across different domains within the same class. We considered a path of interpolation rather than a fixed coefficient for mixing the data. Moreover, we performed nonlinear interpolation in the latent space rather than linear interpolation in the pixel space. We conclude that these reasons would have contributed to the better performance of DNT.

Visualization of the latent space. We visualize the latent representations obtained from DeepAll, Manifold Mixup, and DNT to compare their ability (i) to separate classes of the source domains and (ii) to create invariance between domains. Fig. 5 and Fig. 6 display the 2D view of the latent space $\mathcal{Z}$ plotted via TSNE (Van der Maaten and Hinton, 2008).

We compare the representations learned by DeepAll, Manifold Mixup, and DNT in 2 settings: 100% and 5% training data. We sampled 500 data points for visualization from the validation set and obtained the $\mathbf{z}$ samples by passing them through the encoders of all the models. We use different colors to denote the classes and different marker styles to denote the domains. Figure 5 demonstrates the separation of classes achieved by DeepAll, Manifold Mixup, and DNT in the 100% and 5% training data settings. We can observe that data from different domains are closer for DNT within each class. Moreover, the classes are well separated in the latent space. In the 5% data scenario, the class separation is comparatively less pronounced than in the 100% case. Nonetheless, in both settings, our model DNT performs the separation of classes better than DeepAll and Manifold Mixup.



(a) DeepAll - 100% Data

(b) DeepAll - 5% Data

(c) Manifold Mixup - 100% Data

(d) Manifold Mixup - 5% Data

(e) DNT - 100% Data

(f) DNT - 5% Data

Figure 6: Class-wise distribution of $\mathbf{z}$ from DeepAll, Manifold Mixup, and DNT trained on 100% and 5% data of PACS. Domains are visually well separated for DeepAll and Manifold Mixup but blended together for DNT, indicating better invariance to domain information.

DG techniques that learn domain invariant representations from the data must ignore redundant information from individual domains and focus on the similarity between domains. This can be evaluated based on the similarity between representations of images belonging to the same class but from different domains. A well-learned invariant representation algorithm will map the inputs from various domains to nearby areas in the latent space, provided they are in the same class. We plot the representations of the domains within each class of the PACS dataset in Fig. 6 to observe domain invariance within a class. Fig. 6 displays how the latent representations from different domains lie in each of the 7 classes. We compare DeepAll and Manifold Mixup with DNT in 2 training size settings: 100% and 5%. Each of the 3 source domains can be clearly differentiated between themselves in the DeepAll and Manifold Mixup models. The distinction is more apparent in the case of limited data settings (Fig. 6 (b),(d)). Visualizations from the DNT model, have lesser distinction (more invariance) among the domains than the others. DNT reduces the separation between

the different domains by bringing them closer within a class. This is more evident in the limited data setting. Due to limited data, there is an increase in the variance between the representations $\mathbf{z}$. Interpolation introduces new data around the existing data and can help bridge the variance between representations achieving tighter class-specific representations.

4. Conclusion

We introduced a framework *interpolation robustness* for learning a representation invariant to the interpolation operation across domains. We demonstrated that this representation is robust across domains and, therefore, useful for domain generalization. We also show that the presence of our loss in DIRT (Nguyen et al., 2021) and DGER (Zhao et al., 2020) leads to significantly better performance, especially when data is limited. A key advantage of our proposed framework is that it can be combined with several existing domain generalization methods that use invariant representations to improve their performance.

Acknowledgments

This research was partially supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP210102798). The views expressed herein are those of the authors and are not necessarily those of the Australian Government or Australian Research Council.

References

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2019.

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

Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2224–2233, 2019.

Daniel C Castro, Ian Walker, and Ben Glocker. Causality matters in medical imaging. *Nat Commun*, 11(1):3673, July 2020.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, 2018.

Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*, 2019.

Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders, 2015.

Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, 2019.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189, 2018.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5543–5551, 2017a.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization, 2017b.

Lei Li, Veronika A. Zimmer, Wangbin Ding, Fuping Wu, Liqin Huang, Julia A. Schnabel, and Xiahai Zhuang. Random style transfer based domain generalization networks integrating shape and spatial information, 2020.

Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Massimiliano Mancini, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Best sources forward: domain generalization through source-specific nets, 2018.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation, 2013.

A. Tuan Nguyen, Toan Tran, Yarin Gal, and Atilim Gunes Baydin. Domain invariant representation learning with domain density transformations. In *Advances in Neural Information Processing Systems*, volume 34, pages 5264–5275, 2021.

A. Tuan Nguyen, Thanh Nguyen-Tang, Ser-Nam Lim, and Philip H.S. Torr. Tipi: Test time adaptation with transformation invariance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24162–24171, June 2023.

Alexander Robey, George J Pappas, and Hamed Hassani. Model-based domain generalization. *Advances in Neural Information Processing Systems*, 34:20210–20229, 2021.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states, 2018.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation, 2018.

Shujun Wang, Lequan Yu, Kang Li, Xin Yang, Chi-Wing Fu, and Pheng-Ann Heng. Dofe: Domain-oriented feature embedding for generalizable fundus image segmentation on unseen datasets. *IEEE Transactions on Medical Imaging*, 39(12):4237–4248, 2020a.

Yufei Wang, Haoliang Li, and Alex C. Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, may 2020b. doi: 10.1109/icassp40776.2020.9053273.

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

Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. Improving out-of-distribution robustness via selective augmentation. In *Proceeding of the Thirty-ninth International Conference on Machine Learning*, 2022.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. Mvdg: A unified multi-view framework for domain generalization, 2022.

Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. *Advances in Neural Information Processing Systems*, 33:16096–16107, 2020.

Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation, 2020.

Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization, 2021.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.