# Advancing Domain Generalization through Cross-Domain Class-Contrastive Learning and Addressing Data Imbalances

Thesis submitted in partial fulfillment
of the requirements for the degree of

*Master of Science*
*in*
***Computer Science and Engineering***
*by Research*

by

Saransh Dave
2019701025
saransh.dave@research.iiit.ac.in


International Institute of Information Technology
Hyderabad - 500032, INDIA
December, 2023

International Institute of Information Technology
Hyderabad, India

# CERTIFICATE

It is certified that the work contained in this thesis, titled ***"Advancing Domain Generalization through Cross-Domain Class-Contrastive Learning and Addressing Data Imbalances"*** by Saransh Dave, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Dr. Vineet Gandhi

*To my parents!!!*

# Acknowledgments

I would like to express my sincere gratitude to my advisor, Prof. Vineet Gandhi, for his invaluable guidance and assistance throughout the course of my research. His profound knowledge, patience, and insightful feedback greatly contributed to this work. Prof. Gandhi's unwavering support and commitment have been integral in shaping this research, and I am deeply thankful for his mentorship.

I wish to extend my sincere appreciation to Sarath Sivaprasad, who played an essential role in my research. His talent for simplifying intricate concepts broadened my understanding considerably. Our numerous discussions offered valuable insights that fundamentally influenced my work. Beyond his role as a mentor, Sarath is a supportive friend who is always ready to provide help. His humility, despite his vast expertise, is an attribute I greatly admire. It's fair to say that without Sarath's guidance, my research would not have been possible. For his invaluable contribution, I am deeply grateful.

I am thankful to my family for providing me with their unwavering love and support throughout this journey. Their encouragement and belief in my abilities have been my constant source of strength. I cannot thank them enough for their sacrifices and understanding. Their presence in my life is truly invaluable, and I owe them more than words can express.

I would like to extend my heartfelt appreciation to my hostel friends, Tushar Abhishek, Tushar Chandra, Astitva Srivastava, Ashish Singh, Harsh Shukla, and Shubham Raj. Our time together was filled with laughter, camaraderie, and numerous fun discussions. Their availability and willingness to help whenever needed have been a tremendous support system. I would also like to acknowledge my lab friends, Ritam Basu, Jeet Vora, and Abhinaba Bala. Their enthusiasm for learning and technical discussions added a new dimension to my research journey. Their insights and collaborative efforts greatly enhanced the quality of my work. A special shoutout goes to my math buddies, Prateek Pani and Sai Sumanth Natva. Our shared interest and love for mathematics and puzzles created a dynamic environment that fostered joy and continuous learning. Their presence added vibrant flavors to my research experience.

Lastly, I am grateful to all the mentors, professors, and colleagues who have provided guidance and assistance throughout my research work. Their valuable feedback, suggestions, and expertise have shaped my research and broadened my horizons.

To everyone mentioned above, and to those who have supported me in ways beyond this list, I offer my deepest appreciation. Your contributions have made a significant impact on my academic journey, and I am truly grateful for your presence in my life.

# Abstract

This thesis delves into the critical field of Domain Generalization (DG) in machine learning, where models are trained on multiple source distributions with the objective of generalizing to unseen target distributions. We begin by dissecting various facets of DG, including distribution shifts, shortcut learning, representation learning, and data imbalances. This foundational investigation sets the stage for understanding the challenges associated with DG and the complexities that arise.

A comprehensive literature review is conducted, highlighting existing challenges and contextualizing our contributions to the field. The review encompasses learning invariant features, parameter sharing techniques, meta-learning techniques, and data augmentation approaches.

One of the key contributions of this thesis is the examination of the role low-dimensional representations play in enhancing DG performance. We introduce a method to compute the implicit dimensionality of latent representations, exploring its correlation with performance in a domain generalization context. This essential finding motivated us to further investigate the effects of low-dimensional representations.

Building on these insights, we present Cross-Domain Class-Contrastive Learning (CDCC), a technique that learns sparse representations in the latent space, resulting in lower-dimensional representations and improved domain generalization performance. CDCC establishes competitive results on various DG benchmarks, comparing favorably with numerous existing approaches in DomainBed.

Venturing beyond traditional DG, we discuss a series of experiments conducted for domain generalization in long-tailed settings, which are common in real-world applications. Additionally, we present supplementary experiments yielding intriguing findings. Our analysis reveals that the CDCC approach exhibits greater robustness in long-tailed distributions and that the order of performances across test domains remains unaffected by the order of training domains in the long-tailed setting. This section aims to inspire researchers to further probe the outcomes of these experiments and advance the understanding of domain generalization.

In conclusion, this thesis offers a well-rounded exploration of DG by combining a comprehensive literature review, the discovery of the importance of low-dimensional representations in DG, the development of the CDCC method, and the meticulous analysis of long-tailed settings and other experimental findings.

# Contents

# List of Figures

*Chapter 1*

# Introduction

Neural networks have become an essential tool in the field of machine learning owing to their ability to learn complex patterns and representations from data. In a standard machine learning setting, these networks are trained and evaluated under the assumption that the data is independently and identically distributed (IID). This means that both the training and test data are assumed to be sampled from the same underlying distribution, leading to the expectation that the network's performance on the training data will be a reliable indicator of its performance on the test data.

However, real-world scenarios often deviate from the IID assumption. In practice, test data may originate from a distribution that differs from the one used for training. This discrepancy can result in a significant decrease in a neural network's performance when it is deployed in an actual situation. To address this challenge, researchers have been exploring methods to enhance the robustness of neural networks to distribution shifts, one of which is Domain Generalization (DG).

DG is a framework designed to evaluate a model's robustness in scenarios where the test data comes from an unseen domain. The approach involves training a model using multiple related domains in the training data and assessing its ability to classify the same classes in the test domain. For example, a model may be trained on labeled image data from photos, paintings, and cartoons, focusing on discriminating between specific classes. The model's effectiveness is then tested on its ability to classify those same classes in an entirely different domain, such as sketches (Li et al., 2017) [57].

The DG framework poses a more challenging optimization problem compared to other methods, such as Domain Adaptation (DA). While DA and other similar frameworks rely on certain assumptions about the target distribution, DG does not. Instead, it aims to minimize the expected classification loss across all possible domains that can meaningfully represent the given classes.

However, DG is built upon a strong assumption that data from all classes are available in all domains, which can be difficult to satisfy in many real-world situations. This limitation highlights the ongoing challenges in developing models that are robust to distribution shifts.

To further improve neural network's robustness to distribution shifts, researchers are investigating various techniques, such as domain adaptation, domain alignment, and data augmentation. These methods focus on aligning the feature distributions between the source and target domains or augmenting the

training data to cover a wider range of possible scenarios, ultimately helping the models to better handle non-IID settings.

Understanding and addressing distribution shifts is a critical aspect of developing neural networks that can maintain their performance in real-world applications. Domain Generalization, along with other techniques, contributes to the ongoing efforts in enhancing the robustness of machine learning models, ensuring they can effectively adapt to changing data distributions and deliver reliable results.

With this introduction to Domain Generalization, we will explore the formal definition of DG, various aspects involved in understanding DG, various datasets used in the context of DG for analyzing distribution shifts, and finally, end with understanding some related work in DG literature.

## 1.1 Domain Generalization

Domain generalization is a technique used in machine learning to improve the performance of models on unseen data. In traditional machine learning, models are trained and evaluated on a fixed dataset that is assumed to be representative of the data that the model will encounter in the future. However, in practice, the data that a model encounters in the real world may differ from the training data, leading to poor performance.

Domain generalization aims to address this problem by training models on multiple datasets that cover a wide range of possible scenarios rather than just one fixed dataset. By doing so, the model can learn to generalize its predictions to new data that it has not encountered before. This technique can be particularly useful in situations where the available training data is limited or biased and where the model needs to perform well on a variety of different tasks or in different contexts.

In a DG setting, the datapoints consist of tuples of form $(x, y, d)$ where $x$ is the input feature, $y$ is the label corresponding to to input $x$, and $d$ is the domain corresponding to input $x$. In general, if we consider a set of labels $L$, and a set of domains $D$ to represent a DG dataset $S$ containing $n$ samples, then $S$ can be represented as follows:

$$S = \{(x_i, y_i, d_i)\}_{i \in [1,\ldots,n]} \tag{1.1}$$

Here, $(x_i, y_i, d_i)$ corresponds to the $i^{th}$ datapoint of the dataset. Also, $y_i \in L;\ \forall i \in [1, \ldots, n]$, and $d_i \in D;\ \forall i \in [1, \ldots, n]$. For this dataset $S$, if we create a train-test split such that the training set consists of samples with domain set $D_{train}$ where $D_{train} \subset D$, and the test set consists of samples with domain set $D_{test}$ where $D_{test} = D - D_{train}$. Then the training set $S_{train}$, and the test set $S_{test}$ can be represented as follows:

$$S_{train} = \{(x, y, d) \,|\, (x, y, d) \in S,\, d \in D_{train}\} \tag{1.2}$$

$$S_{test} = \{(x, y, d) \,|\, (x, y, d) \in S,\, d \in D_{test}\} \tag{1.3}$$

In DG, the aim is to train models that can generalize to unseen domains. To achieve this, train-test splits in DG are typically performed such that the training and test sets contain different domains. This shows that:

$$(x_1, y_1, d_1) \in S_{train} \quad \& \quad (x_2, y_2, d_2) \in S_{test} \quad \Longrightarrow \quad d_1 \neq d_2 \tag{1.4}$$

Figure 1.1 shows a possible train test split on some sample images from the PACS dataset. The dataset contains the classes Dog, Elephant, Giraffe, Guitar, Horse, House, and Person, and for every class, it further contains the domains Photo, Art-Painting, Cartoon, and Sketch. The train test split shown in Figure 1.1 contains all samples belonging to the domain Photo, Art-Painting, and Cartoon in the train split and the remaining samples belonging to the Sketch class in the test split.

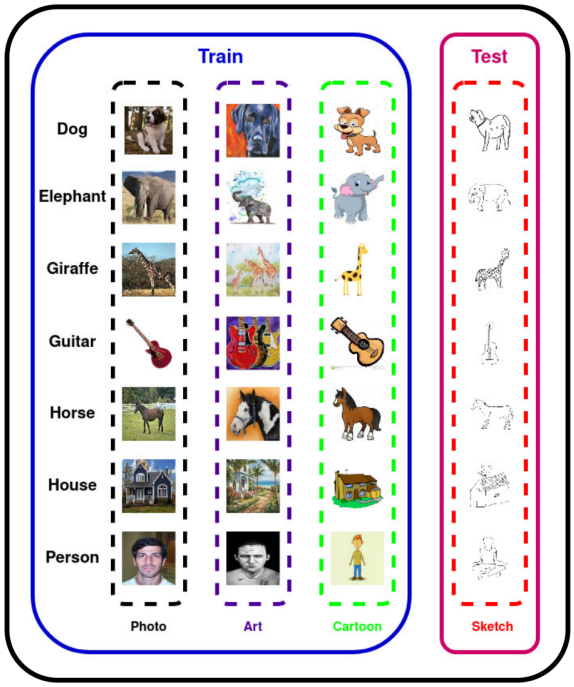



**Figure 1.1** This figure shows one of the four splits for Domain Generalization on some sample images from the PACS dataset.

## 1.2 Several Aspects of Domain Generalization

Domain Generalization (DG) is a rapidly evolving field that aims to address the challenges associated with distribution shifts in machine learning applications. In this section, we delve into the various aspects of DG, exploring the techniques, methodologies, and considerations that researchers and practitioners need to take into account when designing models capable of generalizing across multiple domains. By examining these different facets, we aim to provide a comprehensive understanding of the

current state of the art in DG and highlight the key factors that contribute to the success of domain-generalizing models.

### 1.2.1 Distribution Shift

Distribution shift refers to the phenomenon where the underlying data distribution changes between the training and test phases of a machine learning model. It is a critical challenge in real-world applications, as models that perform well on training data may experience a significant drop in performance when deployed on unseen data with different characteristics. In this section, we provide a descriptive overview of the concept of distribution shift, its various types, and its implications on machine learning models.

Distribution Shift (or Dataset Shift) represents any scenario where the joint probability distribution of the input features and the output labels differ between the training set and the test set. For a training dataset $D_{train}$ and a test dataset $D_{test}$, we say that there is a distribution shift between $D_{train}$ and $D_{test}$, if their corresponding joint probability distributions follow the below relation:

$$P_{train}(x, y) \neq P_{test}(x, y) \tag{1.5}$$

Here, $P_{train}(x, y)$ is the joint probability distributions of the input feature vector $x$ and output label $y$ for the training set, and $P_{test}(x, y)$ is the joint probability distributions of the input feature vector $x$ and output label $y$ for the test set. The datapoints of $D_{train}$ are sampled from the joint distribution $P_{train}(x, y)$ such that $(x', y') \sim P_{train}(x, y);\ \forall\, (x', y') \in D_{train}$. Similarly, the datapoints of $D_{test}$ are sampled from the joint distribution $P_{test}(x, y)$ such that $(x', y') \sim P_{test}(x, y);\ \forall\, (x', y') \in D_{test}$.

There are several types of distribution shifts that can occur in practice. Some of the most common ones are as follows:

- **Covariate Shift**: Covariate shift occurs when the input feature distribution, $p(x)$, changes between the train and test set, but the conditional distribution of labels given the inputs i.e. $p(y|x)$, remains the same. This can be mathematically presented as follows:

  $$P_{train}(x) \neq P_{test}(x) \text{ but } P_{train}(y|x) = P_{test}(y|x) \tag{1.6}$$

  It can be shown that this is a case of distribution shift as follows:

  $$P_{train}(x) \neq P_{test}(x) \text{ and } P_{train}(y|x) = P_{test}(y|x)$$
  $$\implies P_{train}(x)P_{train}(y|x) \neq P_{test}(x)P_{test}(y|x) \tag{1.7}$$
  $$\implies P_{train}(x, y) \neq P_{test}(x, y) \tag{1.8}$$

  Covariate shift is commonly encountered in situations where the data collection process is different for the train and the test sets, or if the environment in which the data is collected changes over

time, leading to different input distributions. For instance, consider a training dataset of photos of cats and dogs taken in an outdoor environment. For this training dataset, if we consider a corresponding test set of photos of cats and dogs taken in a dark indoor environment, this scenario can be considered as an example of covariate shift, as there is a change in the input features across the train and test sets, but the labels given the input features (photos) remain the same.

It is to be noted that the standard Domain Generalization setting falls under the category of covariate shifts as the input features of the training domain differs from the input features of the test domain, implying that $P_{test}(x) \neq P_{train}(x)$. But the output labels (given the input features) remain the same across all the domains, implying that $P_{train}(y|x) = P_{test}(y|x)$.

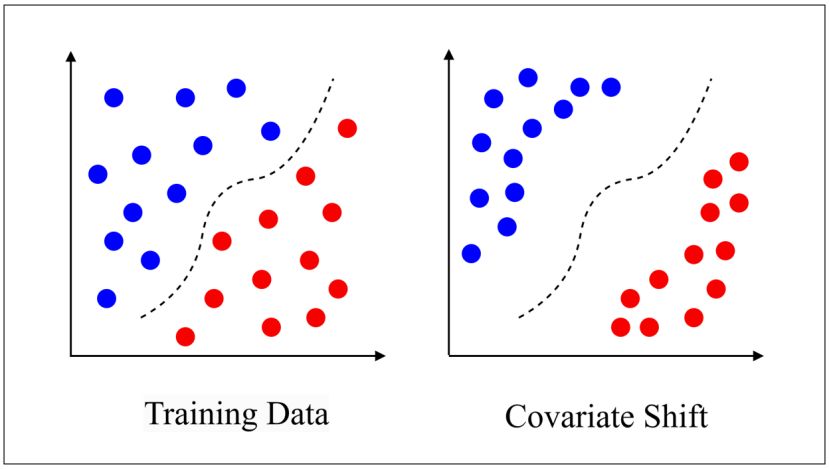



**Figure 1.2** This illustration offers a visual portrayal of covariate shift, emphasizing the changes in the distribution of the input features between the training and test set. But the labels given the input features, stay the same across both.

- **Concept Shift**: Concept shift, also known as concept drift, occurs when the relationship between the input features and the output labels changes between the train and the test set. In this case, the input feature distribution, $p(x)$ for both the training and the test set, remains unchanged, but the conditional distribution of labels given the inputs, $p(y|x)$, can change. This can be mathematically presented as follows:

$$P_{train}(x) = P_{test}(x) \text{ but } P_{train}(y|x) \neq P_{test}(y|x) \tag{1.9}$$

It can be shown that this is a case of distribution shift as follows:

$$P_{train}(x) = P_{test}(x) \text{ and } P_{train}(y|x) \neq P_{test}(y|x)$$
$$\implies P_{train}(x)P_{train}(y|x) \neq P_{test}(x)P_{test}(y|x) \tag{1.10}$$
$$\implies P_{train}(x, y) \neq P_{test}(x, y) \tag{1.11}$$

This type of shift can arise in situations where the underlying criterion of classification changes over time. A classic example to explain concept drift is a machine learning model used for email spam detection. In this scenario, the model is trained to distinguish between spam and non-spam emails based on various features such as the sender's email address, email content, subject line, and other metadata. Initially, the model may perform well in detecting spam emails. However, over time, the definition of a spam email might change, and hence, emails that were previously non-spam might start getting classified as spams and vice-versa.

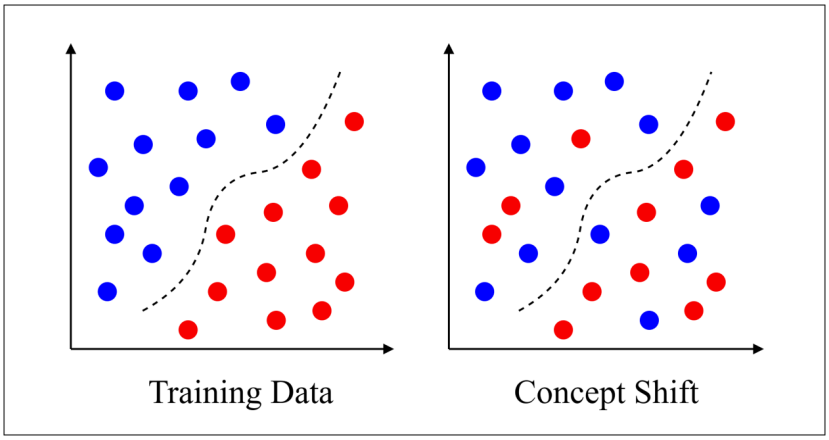



**Figure 1.3** This illustration offers a visual portrayal of concept shift, emphasizing the changes in the relationship between the labels and their corresponding input features across the training and test set, while keeping the distribution of input features unchanged. Note that the position of all the datapoints remain the same across the training and the test set.

- **Label Shift**: Label shift occures when the distribution of output labels changes between the train and the test set, while the conditional distribution of the input features given the label, remain the same. In this case, the output label distribution, $p(y)$, changes between the train and the test set, while the conditional distribution of the input features given the label, $p(x|y)$, remains the same. This can be mathematically presented as follows:

$$P_{train}(y) \neq P_{test}(y) \text{ but } P_{train}(x|y) = P_{test}(x|y) \tag{1.12}$$

It can be shown that this is a case of distribution shift as follows:

$$\begin{aligned} &P_{train}(y) \neq P_{test}(y) \text{ but } P_{train}(x|y) = P_{test}(x|y) \\ &\implies P_{train}(y)P_{train}(x|y) \neq P_{test}(y)P_{test}(x|y) &(1.13)\\ &\implies P_{train}(x,y) \neq P_{test}(x,y) &(1.14)\end{aligned}$$

To illustrate this, consider the task of identifying a disease given a patient's symptoms. Suppose a disease d results in symptoms $s$, which can be represented as $d \rightarrow s$. Regardless of the proportion of patients with disease $d$ in the training and test sets, the relation that a patient with disease $d$ will exhibit symptoms $s$ remains constant across the datasets.

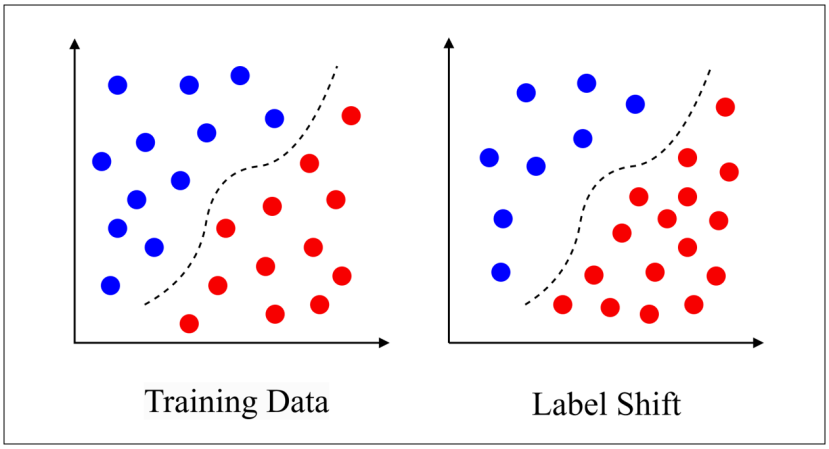



**Figure 1.4** This illustration presents a visual representation of label shift, demonstrating the change in distribution of labels across the training and the test set. It is to be noted that the test set has more of red datapoints and lesser of blue datapoints compared to the training data, hence showing a change in distribution of labels. Also, though it is not visually obvious, the distribution of input features given the label remain the same across the train and the test set.

On a concluding note to this section, the presence of distribution shifts can have a significant impact on the performance and reliability of machine learning models. When a model is trained on a particular distribution and tested on a different one, its performance may degrade considerably. This is because the model has learned patterns specific to the training distribution, which may not generalize well to the test distribution.

Moreover, distribution shifts can lead to biased or unfair predictions in certain cases. For instance, if the training data over-represents certain demographic groups, the model may become biased towards those groups and perform poorly on underrepresented groups when deployed in the real world.

### 1.2.2 Shortcut Learning

Shortcut learning refers to the phenomenon where a machine learning model learns to exploit simpler patterns or correlations in the training data rather than genuinely understanding the underlying concepts it is meant to learn. In the context of classification, shortcut learning occurs in situations where a model ends up learning the subset of features of the input features that are sufficient for performing

classification. If we consider a trained model given by $f()$ that exhibits shortcut learning, then for some input feature vector $x$, the model learns a subset of these input features given by $s(x)$, such that:

$$f(x_1) = f(x_2) \quad if \quad s(x_1) = s(x_2) \qquad (x_1 \neq x_2) \tag{1.15}$$

If it so happens that expected output corresponding to inputs $x_1$ and $x_2$, lets say $y_1$ and $y_2$ respectively, are such that $y_1 = y_2$, but $s(x_1) \neq s(x_2)$, then in this case the model might result in $f(x_1) \neq f(x_2)$ which is a case of misclassification. Similarly, the case when $y_1 \neq y_2$, but $s(x_1) = s(x_2)$ results in misclassification as well. This can lead to models that perform well on the training data but fail to generalize to real-world scenarios or more complex, unseen data.

To illustrate shortcut learning, let's consider the example of training a model to identify cows in images. Suppose the model is provided with a training dataset consisting of images of cows, where the majority of the cows are photographed in grassy fields or on farms. A model that truly understands the concept of a cow would learn to identify its features, such as its shape, the presence of horns, and other characteristics unique to cows, regardless of the background or environment in which the cow is situated.

However, due to shortcut learning, the model might instead focus on the presence of grass or farm-related objects as indicators of a cow being present in the image. This is because the model has learned a spurious correlation between the presence of grass or farm elements and the presence of cows in the training data. The model has essentially taken a "shortcut" in its learning process by relying on these superficial cues rather than learning the intrinsic features of a cow.

Now, let's imagine we present the model with an image of a cow at the beach, a scenario not encountered in the training data. In this situation, the model is likely to fail in identifying the cow, as the image does not contain grass or farm-related elements. The model's reliance on shortcut learning has led to poor generalization performance when faced with an atypical example.

This example highlights the importance of carefully curating training datasets to ensure that models learn to identify the essential features of the target concept rather than relying on simple correlations. It also emphasizes the need for developing algorithms that are robust to shortcut learning, encouraging models to learn the genuine underlying concepts rather than exploiting easy-to-learn patterns in the data.

Figure 1.5 explains another standard example of shortcut learning from [31]. It shows a toy example in which a neural network is trained on a simple dataset of stars and moons located at diffferent positions. A standard neural network can effortlessly categorize new dataset with samples similar to those in the training set (IID). However, when tested on a slightly different dataset (OOD), the network's shortcut strategy is revealed. In this example, the neural network has learned to associate object location with a category. During training, stars were consistently displayed in the top right or bottom left locations of the image, while moons were shown in the top left or bottom right locations. This pattern persists in samples from the IID test set but is absent in OOD test images, which uncovers the shortcut taken by the network, resulting in misclassification.

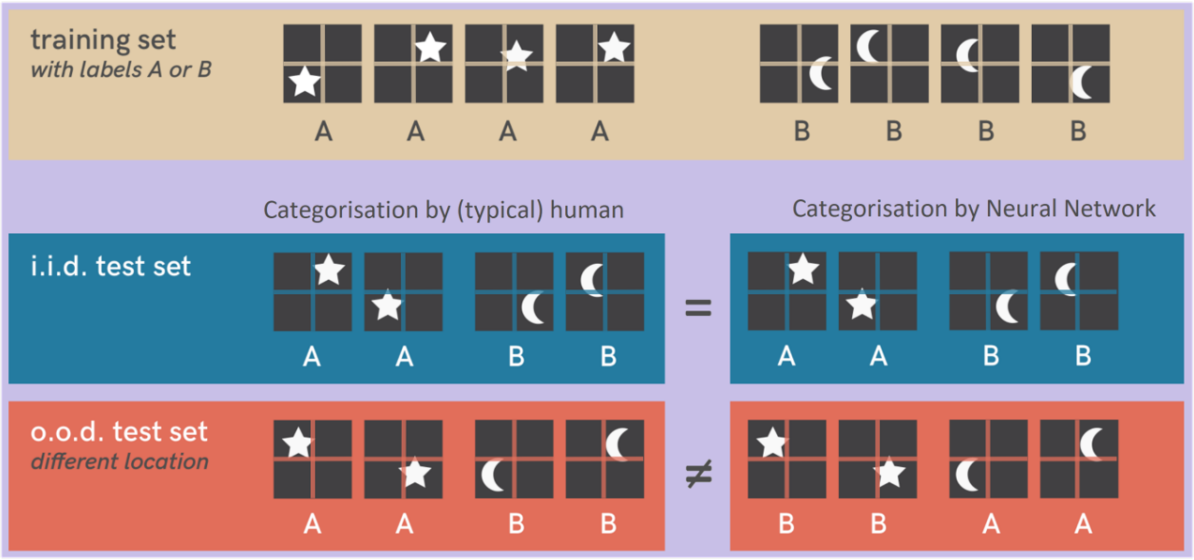



**Figure 1.5** This image shows a toy dataset of starts and moons located at different positions. It shows how a Neural Networks classifies the IID test set samples correctly, but due to shortcut learning, it does not correctly classify the OOD test samples.

(Image Source: [31])

In a broader, philosophical sense, shortcut learning raises questions about the nature of intelligence and understanding. While machine learning models can achieve impressive results in many tasks, their reliance on shortcut learning may indicate a lack of genuine understanding of the concepts they are learning. Developing models that can learn and reason more like humans without relying on shortcut learning remains a challenging and intriguing area of research in artificial intelligence.

### 1.2.3 Representation Learning

Representation learning is at the core of domain generalization in image classification, as it involves automatically discovering and extracting useful and meaningful features from raw image data. These learned representations capture the inherent structure, relationships, and patterns present in images, enabling the model to generalize effectively to new, unseen domains. In the context of domain generalization, the goal is to learn transferable and invariant features across different domains, which makes the model more robust and less sensitive to changes in the data distribution between the source (training) and target (test) domains.

Deep learning architectures, such as convolutional neural networks (CNNs), have been pivotal in representation learning for image classification, as they can automatically learn hierarchical features

from raw data. However, recent advances in unsupervised and self-supervised learning have further broadened the scope of representation learning. One such self-supervised method is contrastive learning, which aims to learn useful representations by comparing and contrasting different features of the data. By encouraging the model to learn representations that bring similar images closer together and push dissimilar images farther apart in the embedding space, contrastive learning allows the model to discover transferable and discriminative features that are essential for domain generalization. By employing techniques like contrastive learning, models can gain a deeper understanding of the underlying structure of image data, allowing them to adapt more efficiently to new, unseen domains. Let's explore Supervised Contrastive Learning in further detail.

- **Supervised Contrastive Learning:**
  Supervised Contrastive Learning (SupCon) [51] is a learning framework that extends the concept of contrastive learning, which has been widely adopted for self-supervised learning (SimCLR [17]), to the supervised setting. In contrastive learning, the goal is to learn representations by contrasting positive and negative examples. In SupCon, distorted pairs are generated by applying different augmentations to the same input instance. This process encourages the model to learn representations that are similar to the augmented versions of the same instance. By doing so, the model is guided to focus on the essential features that remain consistent across various augmentations, further improving its generalization capabilities.

  In Supervised Contrastive Learning, positive pairs consist of any two samples that belong to the same class, indicating that they share similar properties or features. On the other hand, negative pairs are defined as any two samples that belong to distinct classes, implying that they possess different characteristics or features. In the context of contrastive learning, positive and negative pairs play a crucial role in learning meaningful and discriminative representations by encouraging the model to maximize the similarity between positive pairs while minimizing the similarity between negative pairs.

  To understand the Supervised Contrastive Loss ($\mathcal{L}_{SupCon}$), let's consider one mini-batch with $N$ datapoints, we have $N$ tuples of the form (input sample, class label), $(x_k, y_k)_{k=1...N}$. In each input batch of the model, we augment the input to create a pair of two distorted images with the same class labels. We use only the augmented pair (not the actual samples), making the batch size $2N$. The effective dataset for training hence comprises of $2N$ tuples: $(\tilde{x}_l, \tilde{y}_l)_{l=1...2N}$, where $\tilde{x}_{2k}$ and $\tilde{x}_{2k-1}$ are two random augmentations of $x_k$ $(k = 1...N)$ and $\tilde{y}_{2k-1} = \tilde{y}_{2k} = \tilde{y}_k$. Using these conventions, the loss $\mathcal{L}_{SupCon}$ is given as follows,

$$\mathcal{L}_{SupCon} = \sum_{i \in \{1...2N\}} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \bigodot z_p/\tau)}{\sum_{a \in A(i)} \exp(z_i \bigodot z_a/\tau)} \tag{1.16}$$

  Where, the $\bigodot$ denotes the inner (dot) product, $\tau \in \mathcal{R}^+$ is a scalar temperature parameter, and for $i \in \{1...2N\}$:

  - $z_i$ is the Normalized Embedding Vector of $\tilde{x}_i$ (Note that $||z_i|| = 1$)

- $A(i) \equiv \{1...2N\}$ excluding the $i^{th}$ sample
- $P(i) \equiv \{p \in A(i) \,|\, \tilde{y}_p = \tilde{y}_i\}$

Here, $P(i)$ is the set of all samples belonging to the same label as that of the $i^{th}$ sample except for the $i^{th}$ sample itself. In other words, $P(i)$ is the set of all *positives* of the $i^{th}$ sample.

To understand equation 1.16 better, let us try to understand when the loss $\mathcal{L}_{SupCon}$ is minimized. As the component $\log \frac{\exp(z_i \bigodot z_p/\tau)}{\sum_{a \in A(i)} \exp(z_i \bigodot z_a/\tau)}$ of the equation is multiplied with a $-ve$ value, our objective is to maximize this component to so as to minimize the overall expression. For the normalized embedding $z_i$ of the $i^{th}$ sample, this component is maximized when its numerator $\exp(z_i \bigodot z_p/\tau)$ is maximized, and its denominator $\sum_{a \in A(i)} \exp(z_i \bigodot z_a/\tau)$ is minimized. This implies that $(z_i \bigodot z_p)$ should be maximized $\forall p \in P(i)$, i.e. the set of all positive samples, and $(z_i \bigodot z_a)$ should be minimized $\forall a \in A(i) - P(i)$, i.e. the set of all negative samples.

Visually speaking, $(z_i \bigodot z_p)$ is maximized when the embedding unit vectors $z_i$ and $z_p$ are close to each other in a unit hypershpere (ideal case is when $z_i = z_p$). Similarly, $(z_i \bigodot z_a)$ is minimized when the embedding unit vectors $z_i$ and $z_a$ are far from each other in a unit hypershpere (ideal case is when $z_i = -z_a$). See Figure 1.6.

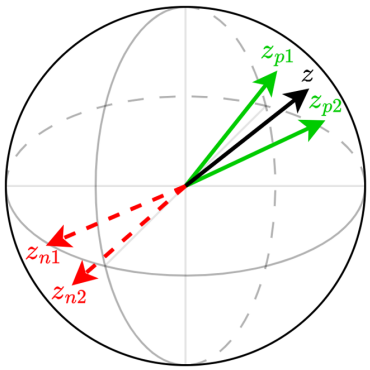



**Figure 1.6** This figure shows a visual of how the normalized embeddings would lie on a unit hypere-sphere after Supervised Contrastive Training. Assumung that $z$ is the embedding for some input samples $s$, then $z_{p1}$ and $z_{p2}$ are the embeddings corresponding to the positives of sample $s$ (hence closer to $z$), and $z_{n1}$ and $z_{n2}$ are the embeddings corresponding to the negatives of sample $s$ (hence farther from $z$).

In Supervised Constrastive Learning (SupCon), data augmentations are employed to create positive pairs and improve the learned representation's robustness and generalization. The augmentations used in SupCon typically include:

**1. Random cropping and resizing:** The input images are cropped randomly and resized to the original dimensions. This augmentation enforces the model to learn features that are invariant to translation and scale changes.

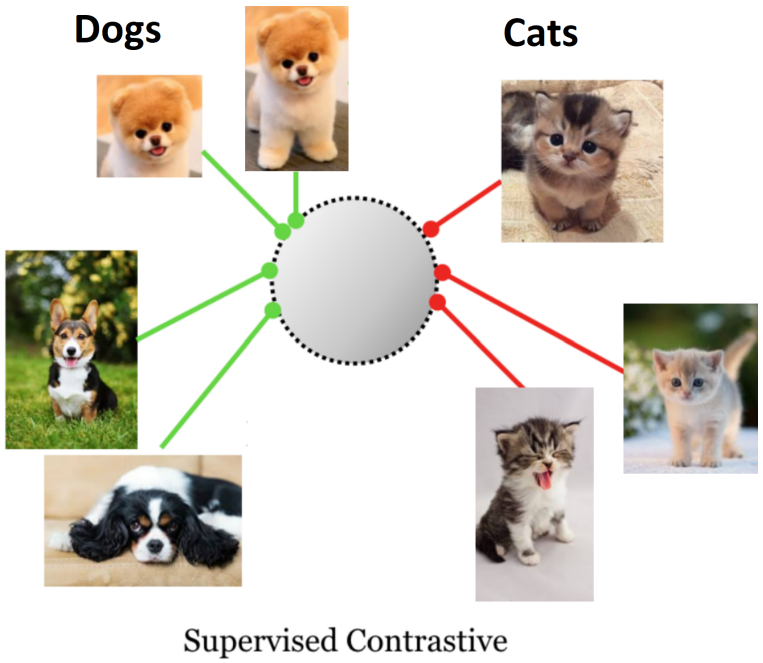



**Figure 1.7** This figure shows a visualization of the learnt Normalized Embeddings with SupCon. It can be seen how the two classes (Dogs and Cats) are clustered such that samples belonging to the same class are closer, and those belonging to the classes are far away from each other.

**2. Random horizontal flipping:** Images are horizontally flipped with a certain probability. This transformation encourages the model to learn features invariant to left-right orientation changes.

**3. Random color jittering:** Randomly perturbing the brightness, contrast, saturation, and hue of input images introduces variations in the color space. This augmentation helps the model to learn features that are robust against color-related variations.

**4. Random grayscale conversion:** With a certain probability, input images are converted to grayscale. This augmentation forces the model to focus on texture and structural features rather than relying on color information.

**5. Gaussian blur:** Applying Gaussian blur with random kernel sizes to the images simulates the effect of out-of-focus images or different camera quality. This transformation encourages the model to learn features invariant to blur variations.

**6. Cutout or Random Erasing:** Randomly masking a portion of the input image with a rectangular patch enforces the model to learn features based on partial information, making it more robust against occlusion.

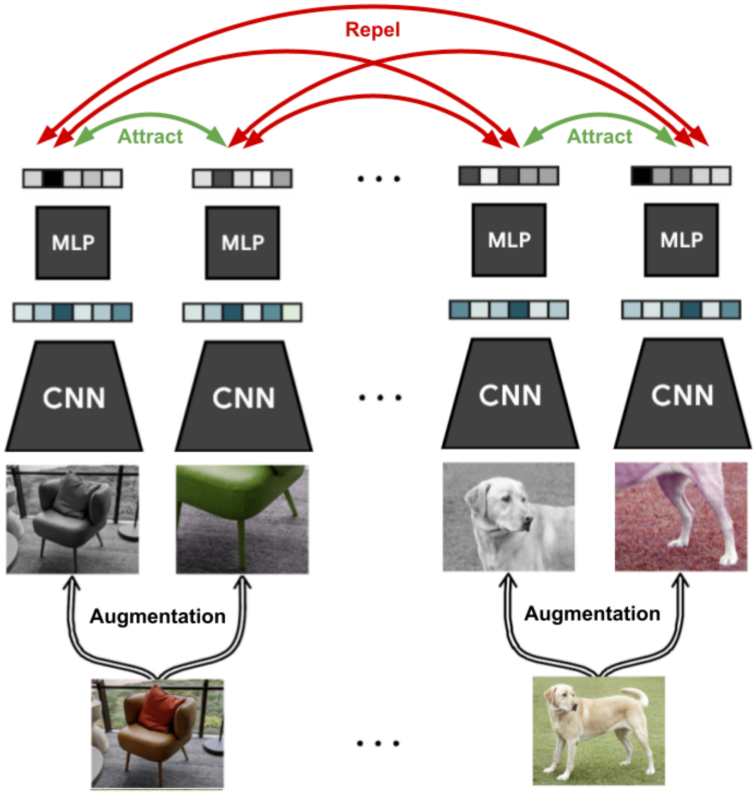



**Figure 1.8** This figure illustrates how SupCon transforms $N$ samples into $2N$ samples using data augmentation and demonstrates the attraction or repulsion between positive and negative samples based on their class membership relative to the anchor. In other words, samples belonging to the same class as the anchor are attracted, while those from different classes are repelled.

These augmentations, when applied in combination, provide a diverse set of positive pairs and help the model learn more robust and generalizable representations. However, the choice of augmentations can vary depending on the specific task and dataset, and it is essential to select relevant and meaningful augmentations to the problem at hand.

We will see more about Contrastive Learning for representation learning in Chapter 3.

### 1.2.4 Data Imbalances

The input data being highly unbalanced results in the model to learn some features with a higher bias compared to others. As most of the real-world datasets follow a long-tailed distribution, it is required for us to come up with better ways of feature learning which are agnostic to data imbalances.

In a domain generalization setting, long-tail data imbalances can manifest in several ways, impacting machine learning model's performance and generalization capabilities. Let's examine the different forms of long-tail imbalances that can occur in domain generalization:

1. **Across classes**: Within a single domain, certain classes may have a significantly larger number of samples compared to others. This class imbalance can lead to models prioritizing learning patterns from the majority classes while neglecting the minority classes. As a result, the model's performance on minority classes may suffer when encountering new, unseen data from the same domain.

2. **Across domains**: There might exist a long-tail distribution across domains, where some domains having significantly more samples compared to others domains. This inter-domain imbalance can pose challenges for models attempting to generalize across multiple domains, as the feature learning gets biased towards the domains with more samples, resulting in poor feature learning from the domains with fewer samples.

3. **Across both classes and domains**: In a more complex scenario, long-tail data imbalances can occur simultaneously across both classes and domains. This can compound the challenges faced by models, as they must not only account for imbalances within each domain but also adapt to the varying number of samples across domains. This makes generalizing to new domains and maintaining performance on minority classes and domains more difficult.

Addressing long-tail data imbalances in a domain generalization setting involves developing techniques that can effectively learn from imbalanced data across multiple domains. Some strategies to tackle these challenges include:

1. **Re-sampling techniques**: Balancing the representation of classes across domains by oversampling minority classes, undersampling majority classes, or a combination of both, potentially using domain-specific re-sampling strategies.

2. **Transfer learning and meta-learning**: Leveraging knowledge from related tasks or domains to improve learning on the long-tailed distribution or adapting to new domains with underrepresented classes.

3. **Loss function modifications**: Designing cost-sensitive learning approaches or customized loss functions that account for class and domain imbalances, encouraging models to prioritize learning from underrepresented samples.

4. **Ensemble methods**: Combining multiple models or learning strategies that focus on improving performance on minority classes and generalizing across domains.

5. **Representation Learning**: Recent advancements in Long-Tail classification demonstrate that utilizing self-supervised learning techniques to enhance representations within the latent space can

significantly boost performance in a long-tail classification setting. The approach involves first applying self-supervised learning to develop improved, and ideally sparse, feature embeddings in the latent space. When these sparse feature representations are then used for classification, it leads to better performance in long-tailed classification settings. Wang et al. (2021) [96] serves as an excellent illustration of how better representation learning contributes to progress in addressing long-tail classification challenges.

By understanding the nuances of long-tail data imbalances across classes, domains, and both, practitioners can develop more robust and generalizable machine learning models that can effectively adapt to diverse real-world scenarios.

## 1.3 Datasets for Studying Distribution Shift

This section provides an overview of various datasets used for studying distribution shifts in machine learning, including DomainBed [36], ImageNet Rendition [39], ImageNet Sketch [93], and WILDS [53] datasets. Understanding distribution shifts is crucial for developing robust and generalizable models that can perform well under different real-world conditions.

### 1.3.1 DomainBed

DomainBed is a benchmark suite designed for evaluating domain generalization algorithms. It contains several datasets with multiple domains, allowing researchers to train and evaluate algorithms that can generalize across these domains. The datasets included in DomainBed are:

- **Rotated MNIST (R-MNIST)**: R-MNIST [33] is a variant of the MNIST handwritten digit dataset, where the digits are rotated by different angles in each domain.

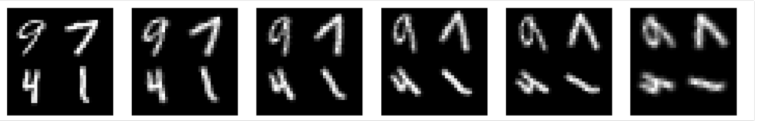



**Figure 1.9** This figure shows some samples from the R-MNIST dataset where the digits are rotated at different angles.

- **Colored MNIST (C-MNIST)**: C-MNIST [5] is a variant of the MNIST dataset, where the digits are colored using different color schemes across domains. This dataset introduces a distribution shift by altering the color of the digits, which provides a unique challenge for domain generalization algorithms.

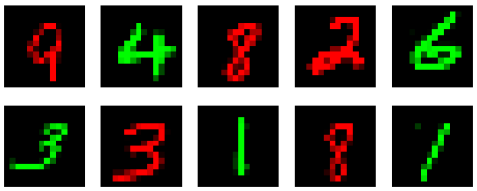


**Figure 1.10** This figure shows some samples from the C-MNIST dataset where the digits are colored in either Red or Green.

- **PACS (Photo, Art, Cartoon, Sketch)**: PACS [57] is a dataset containing images from four different domains (photographs, art, cartoons, and sketches), each with the same seven object classes. This is the most widely used dataset for performing initial experiments for any DG task.

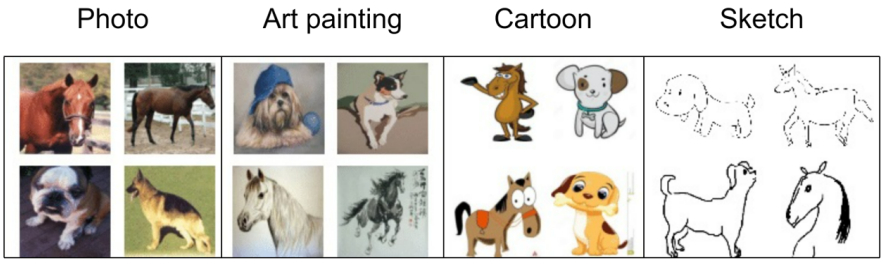



**Figure 1.11** This figure shows some samples from the PACS dataset.

- **VLCS (Pascal VOC2007, LabelMe, Caltech-101, SUN09)**: VLCS [26] is a dataset of images from four different datasets, each representing a different domain, with five common object classes.

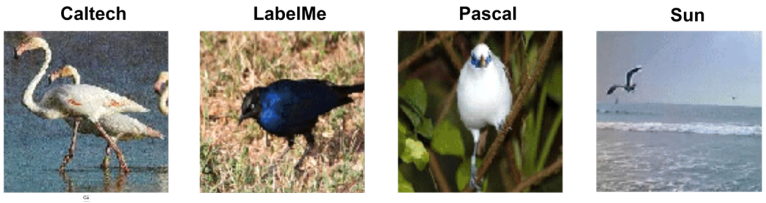



**Figure 1.12** This figure shows some samples from the VLCS dataset. All these samples belong to class bird.

- **Office-Home**: Office-Home [91] is a dataset consisting of images from four distinct domains (Art, Clipart, Product, and Real-World), each containing images of 65 object classes.


**Figure 1.13** This figure shows some samples from the OfficeHome dataset.

- **DomainNet**: DomainNet [74] is a large-scale benchmark for domain generalization, containing six different domains: Clipart, Infograph, Painting, Quickdraw, Real, and Sketch. Each domain consists of images from 345 object classes.

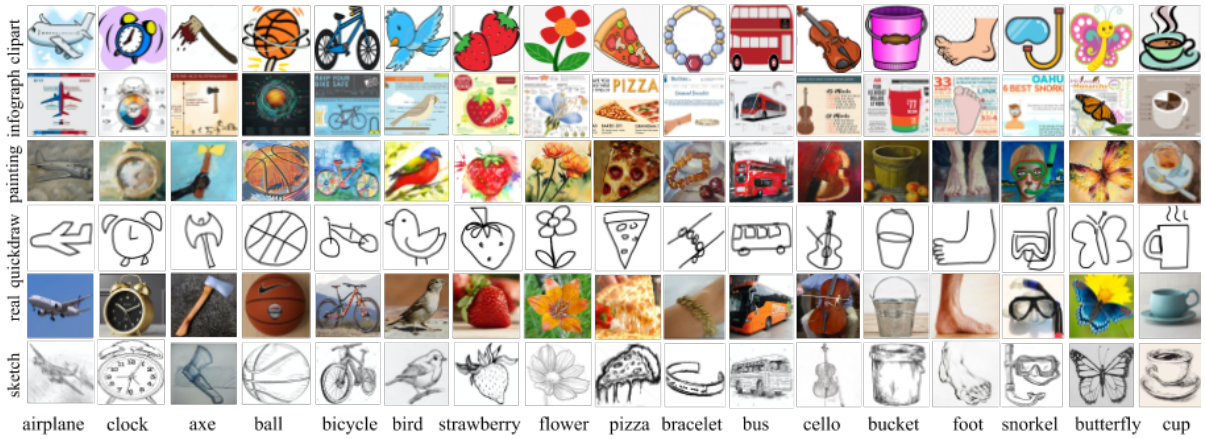



**Figure 1.14** This figure shows some samples from the DomainNet dataset.

- **TerraIncognita**: TerraIncognita [10] is a dataset consisting of images featuring wild animals taken from various locations. It includes four domains (L100, L38, L43, and L46) and consists of 10 different categories that are common across all the domains.

We will explore more about these datasets in Chapter 3.

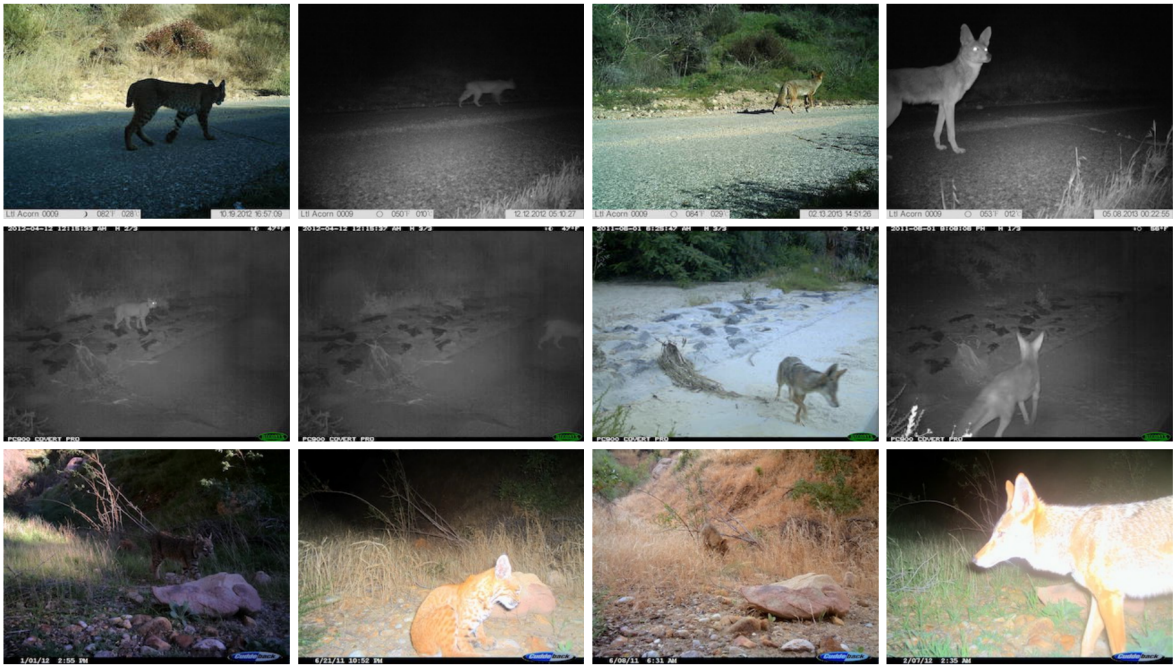


**Figure 1.15** This figure shows some samples from the TerraIncognita dataset. Every row in this figure corresponds to a set of images taken from a particular location.

### 1.3.2 ImageNet Rendition

ImageNet Rendition is a dataset derived from the original ImageNet dataset. It introduces distribution shifts by modifying the images using various image processing techniques, such as changes in color balance, saturation, contrast, the addition of noise, blurring, and the application of artistic filters. It contains art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, and video game renditions of a subset of 200 ImageNet classes. This dataset allows researchers to study the effects of distribution shift on model performance and robustness due to changes in visual style.

### 1.3.3 ImageNet Sketch

ImageNet Sketch is another dataset derived from the original ImageNet dataset. It consists of hand-drawn sketches representing the same object categories as the original dataset. It consists of 50 images for each of the 1000 ImageNet classes (i.e., 50,000 images in total). Sketches inherently introduce a significant distribution shift, as they are abstract, simplified, and stylistically diverse compared to photographs. This dataset enables researchers to investigate a model's ability to generalize to different visual representations and identify potential limitations.

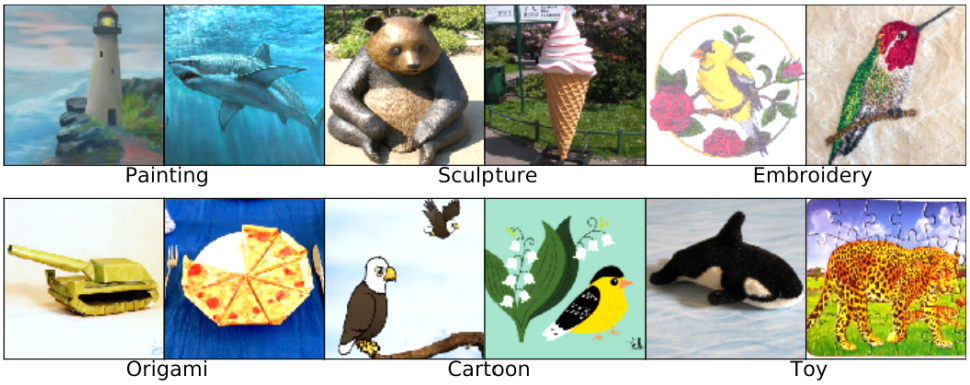



**Figure 1.16** This figure shows some samples from the ImageNet-Rendition dataset.

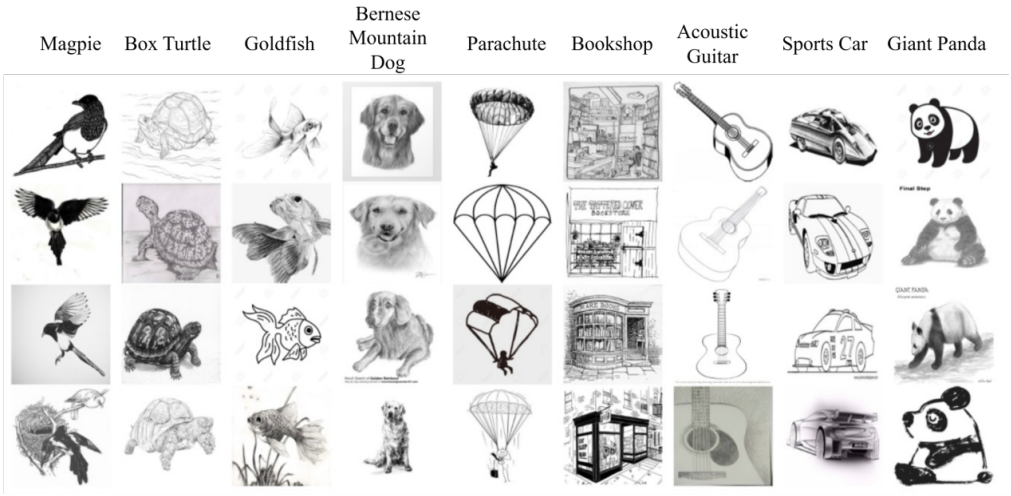



**Figure 1.17** This figure shows some samples from the ImageNet-Sketch dataset.

### 1.3.4 WILDS

WILDS is a collection of large-scale datasets designed to study the problem of distribution shifts in various real-world settings. Each dataset in WILDS comes with a natural distribution shift between the training and test sets. The datasets included in WILDS are:

- **iWildCam**: iWildCam [9] is a dataset for animal species classification in camera trap images, with distribution shifts due to different camera trap locations and camera intrinsics.

- **Camelyon17**: Camelyon17 [8] is a dataset for metastatic cancer detection in histopathology images, with distribution shifts due to data from different hospitals.

- **RxRx1**: RxRx1 [89] is a cell biology dataset with microscopy images of human cells subjected to genetic perturbations. The dataset contains batch-to-batch variations, which introduce distribution shifts.
- **FMoW**: FMoW [19] is a dataset for land use classification in satellite images, with distribution shifts due to changes in regions and images across time.
- **PovertyMap**: PovertyMap [101] is a dataset for predicting economic well-being from satellite imagery, with distribution shifts due to variations in location, both due to country-to-country variations and also due to variations in rural and urban regions within a country.
- **GlobalWheat**: GlobalWheat [20] [21] is a dataset for wheat head detection in agricultural images, with distribution shifts due to geographic location and variation across time.
- **OGB-MolPCBA**: OGB-MolPCBA [43] is a dataset designed for predicting molecular properties, exhibiting distribution shifts resulting from the utilization of distinct molecule scaffolds.
- **CivilComments**: CivilComments [13] is a dataset of online comments annotated for toxicity, with distribution shifts arising from different discussion groups and comment sources.
- **Amazon**: The Amazon dataset [73] is a collection of product reviews used for user sentiment analysis. The goal is to predict the sentiment of a user based on the text of their review. Distribution shifts may occur due to varying user preferences.
- **Py150**: Py150 [64] [80] is a dataset for predicting which Python functions contain bugs based on their source code, with distribution shifts due to different repositories and programming styles.

The discussed datasets, consisting of DomainBed, ImageNet Rendition, ImageNet Sketch, and WILDS, provide a comprehensive set of resources for researchers to study distribution shifts and develop machine learning models that are more robust and generalizable across different conditions.

## 1.4 Thesis Contributions

The main contributions of this thesis are threefold. First, we identify the importance of low-dimensional representations for domain generalization performance. We introduce a method to compute the implicit dimensionality of latent representations, exploring its correlation with performance in a domain generalization context. This essential finding motivated us to further investigate the effects of low-dimensional representations.

Second, we present Cross-Domain Class-Contrastive Learning (CDCC), a technique that learns sparse representations in the latent space, resulting in lower-dimensional representations and improved domain generalization performance. CDCC establishes competitive results on various DG benchmarks, comparing favorably with numerous existing approaches.

| | Dataset | iWildCam | Camelyon17 | RxRx1 | FMoW | PovertyMap | GlobalWheat | OGB-MolPCBA | CivilComments | Amazon | Py150 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Input (x) | camera trap photo | tissue slide | cell image | satellite image | satellite image | wheat image | molecular graph | online comment | product review | code |
| | Prediction (y) | animal species | tumor | perturbed gene | land use | asset wealth | wheat head bbox | bioassays | toxicity | sentiment | autocomplete |
| | Domain (d) | camera | hospital | batch | time, region | country, ru/ur | location, time | scaffold | demographic | user | git repo |
| | Source example |  |  |  |  |  |  |  | What do Black and LGBT people have to do with bicycle licensing? | Overall a solid package that has a good quality of construction for the price. | `import numpy as np … norm=np.___` |
| | Target example |  |  |  |  |  |  |  | As a Christian, I will not be patronizing any of those businesses. | I *loved* my French press, it's so perfect and came with all this fun stuff! | `import subprocess as sp p=sp.Popen() stdout=p.___` |
| | Original paper | Beery et al. 2020 | Bandi et al. 2018 | Taylor et al. 2019 | Christie et al. 2018 | Yeh et al. 2020 | David et al. 2021 | Hu et al. 2020 | Borkan et al. 2019 | Ni et al. 2019 | Raychev et al. 2016 |
| Labeled | # domains | 323 | 5 | 51 | 16 x 5 | 23 x 2 | 47 | 120,084 | 16 | 3,920 | 8,421 |
| | # examples | 203,029 | 455,954 | 125,510 | 141,696 | 19,669 | 6,515 | 437,929 | 448,000 | 539,502 | 150,000 |
| Unlabeled – Source domains | # domains | - | 3 | - | 11 x 5 | 13 x 2 | 18 | 44,930 | - | - | - |
| | # examples | - | 1,799,247 | - | 11,948 | 181,948 | 5,997 | 4,052,627 | - | - | - |
| Unlabeled – Extra domains | # domains | 3,215 | - | - | - | - | 53 | - | 1 | 21,694 | - |
| | # examples | 819,120 | - | - | - | - | 42,445 | - | 1,551,515 | 2,927,841 | - |
| Unlabeled – Validation domains | # domains | - | 1 | - | 3 x 5 | 5 x 2 | 11 | 31,361 | - | 1,334 | - |
| | # examples | - | 600,030 | - | 155,313 | 24,173 | 2,000 | 430,325 | - | 266,066 | - |
| Unlabeled – Target domains | # domains | - | 1 | - | 2 x 5 | 5 x 2 | 18 | 43,793 | - | 1,334 | - |
| | # examples | - | 600,030 | - | 173,208 | 55,275 | 8,997 | 517,048 | - | 268,761 | - |

**Figure 1.18** This figure shows a summary of all the datasets from WILDS. It summarizes the input, output and domain types for each of the datasets along with a sample example for each. It also consists of metadata like total number of domains, total number of samples and spilt of the samples across the domains for all the datasets. (Image Source: https://wilds.stanford.edu/datasets/)

Third, we discuss a series of experiments conducted for domain generalization in long-tailed settings, which are common in real-world applications. Additionally, we present supplementary experiments yielding intriguing findings. Our analysis reveals that the CDCC approach exhibits greater robustness in long-tailed distributions and that the order of performances across test domains remains unaffected by the order of training domains in the long-tailed setting. This section aims to inspire researchers to further probe the outcomes of these experiments and advance the understanding of domain generalization.

To summarize it all, this thesis makes the following contributions:

- Examination of the role of low-dimensional representations in enhancing DG performance.
  - Introduced a method to compute the implicit dimensionality of latent representations.
  - Explored the correlation between implicit dimensionality and performance in a domain generalization context.
- Presentation of Cross-Domain Class-Contrastive Learning (CDCC).
  - A technique that learns sparse representations in the latent space.

  - Results in lower-dimensional representations and improved domain generalization performance.
  - Establishes competitive results on various DG benchmarks.
- Findings about domain generalization in long-tailed settings.
  - CDCC approach exhibits greater robustness in long-tailed distributions.
  - Order of performances across test domains remains unaffected by the order of training domains.

These contributions have the potential to advance the state-of-the-art in domain generalization and make it more applicable to real-world problems.

## 1.5 Conclusion

In conclusion, this introductory chapter has provided a comprehensive overview of domain generalization, an important and rapidly growing area in machine learning research. We have delved into the various aspects and challenges of domain generalization, shedding light on the complexities involved in creating models that can generalize effectively across different domains and tasks.

We have discussed the concept of distribution shift, which arises when the data distribution in the target domain differs from the training data distribution and explored shortcut learning, where models exploit superficial correlations in the training data, leading to poor generalization. Representation learning has been highlighted as a key component in achieving domain generalization, focusing on learning features that are invariant across different domains. Additionally, we have addressed the issue of data imbalances, which can further hinder the generalization performance of models.

In order to study distribution shifts and evaluate the performance of domain generalization methods, we have explored several benchmark datasets, including DomainBed, ImageNet Rendition, ImageNet Sketch, and WILDS. These datasets provide diverse challenges and unique opportunities for the development and testing of new domain generalization techniques, allowing researchers to push the boundaries of current AI capabilities.

As we move forward, the foundations laid in this chapter will serve as a stepping stone for a deeper understanding of the complexities and subtleties of domain generalization and its broader applications in the coming chapters. Addressing the challenges and opportunities highlighted in this chapter will be crucial in developing more robust and generalizable models for real-world applications.

The flow of further chapters is as follows,

- **Chapter 2: Related Literature**, This chapter reviews seminal literature and research, identifying the existing challenges and setting the context for understanding our contributions to the field.

- **Chapter 3: Cross-Domain Class-Contrastive Learning: Finding Lower Dimensional Representations for Improved Domain Generalization**, This chapter serves as the primary contribution of this thesis. In this chapter, we propose a novel method to compute the implicit dimensionality of latent representations and also study its correlation with performance in a domain generalization setting. Taking forward the findings of this study, we propose a new Cross-Domain Class-Contrastive Learning (CDCC), which learns a sparse representation in the latent space and hence results in a better domain generalization performance.

- **Chapter 4: Going beyond Traditional Domain Generalization**, In this chapter, we discuss various experiments performed for domain generalization in a long-tailed setting, as well as some additional experiments that resulted in some interesting findings. This chapter is intended to motivate researchers in the field to further investigate the findings of these experiments and take forward the understanding of domain generalization.

- **Chapter 5: Conclusion**, This chapter finally provides some concluding remarks on the overall work stated in this thesis.

*Chapter 2*

# Related Literature

The field of domain generalization has attracted significant attention in recent years, with numerous approaches being proposed to address the challenges associated with learning invariant features, sharing parameters, utilizing meta-learning techniques, and applying data augmentation. This chapter provides a comprehensive review of the related literature, focusing on seminal works and the key advancements that have shaped the current state of domain generalization research. We begin by discussing the progress in learning invariant features, followed by parameter sharing techniques, meta-learning strategies, and data augmentation approaches.

## 2.1 Learning Invariant Features

Invariant features are features that are not affected by changes in the environment or the way the data is presented. For example, an invariant feature for a face recognition system might be the shape of the nose, which would be the same regardless of the person's pose or lighting conditions. In the context of machine learning, invariant features are those features that remain unchanged despite variations in the input data for a given label. Learning invariant features is a challenging problem, but it is essential for many machine learning tasks. For example, in a domain generalization setting, where the model is trained on data from a variety of domains and then tested on data from a new domain, invariant features can help the model to generalize to the new domain.

There are a number of different approaches to learning invariant features. One common approach is to use feature extraction algorithms, which extract features from the data that are invariant to certain types of changes. For example, a feature extraction algorithm might extract features that are invariant to rotation, translation, or scale.

Another approach to learning invariant features is to use deep learning algorithms. Deep learning algorithms can learn to extract invariant features from data by training on a large dataset of labeled data. The choice of approach to learning invariant features depends on the specific machine learning task. However, in general, invariant features can be a valuable asset for machine learning models.

In the context of previous works on learning invariant features for improving domain generalization, Kernel methods have been employed to identify a feature transformation that simultaneously minimizes the distance between transformed feature distributions across domains and preserves the information between the original features and targets [70]. In their groundbreaking work, Ganin et al. [30] introduced Domain Adversarial Neural Networks (DANN), a domain adaptation method that leverages Generative Adversarial Networks (GANs) [34] to learn a feature representation consistent across training domains.

DANN has been extended by Akuzawa et al. [2] to address cases where a statistical dependence exists between domain and class label variables. Albuquerque et al. [3] further developed DANN by considering one-versus-all adversaries attempting to predict each example's training domain. Li et al. [60] applied GANs and the maximum mean discrepancy criteria [35] to align feature distributions among domains.

Matsuura and Harada [67] utilized clustering techniques to learn domain-invariant features when training domain separation is unknown. Li et al. [61, 62] discovered a feature transformation $\phi$ ensuring that the transformed input $\phi(x)$, given the output label $y$, belong to the same distribution for all the training domains. Shankar et al. [83] employed a domain classifier to create adversarial examples for a label classifier and vice versa, resulting in an improved label classifier for domain generalization.

By training a robust feature extractor and classifier, Li et al. [59] achieved robustness by (i) requiring the feature extractor to produce features that allow a classifier trained on domain $d$ to classify instances for domain $d' \neq d$, and (ii) demanding the classifier to predict labels on domain $d$ using features generated by a feature extractor trained on domain $d' \neq d$. Li et al. [56] implemented a lifelong learning approach to address the domain generalization problem. Motiian et al. [69] learned a feature representation that satisfied three conditions: (i) examples from different domains with the same class are close, (ii) examples from different domains and classes are distant, and (iii) training examples can be accurately classified.

Ilse et al. [47] trained a variational autoencoder [52] to factorize the bottleneck representation knowledge about domain, class label, and residual variations in the input space. Fang et al. [26] learned a structural SVM metric to guarantee that each example's neighborhood includes examples from the same category and all training domains. The algorithms of Sun and Saenko [87], Sun et al. [86], and Rahman et al. [78] matched feature covariance (second-order statistics) across training domains at some representation level. Ghifary et al. [32] and Hu et al. [42] employed kernel-based multivariate component analysis to reduce the mismatch between training domains while maximizing class separability.

Despite its popularity, learning domain-invariant features has been criticized [106, 49]. There exist alternative approaches, as discussed below. Peters et al. [75] and Rojas-Carulla et al. [81] suggest searching for features that yield the same optimal classifier across training domains. In their pioneering work, Peters et al. [75] connected this type of invariance to the causal structure of data and proposed a fundamental algorithm to learn invariant linear models based on feature selection. Arjovsky et al. [5] expanded upon the previous work to include general gradient-based models, such as neural networks, in their Invariant Risk Minimization (IRM) principle. Teney et al. [90] built upon IRM to learn a

feature transformation that minimizes the relative variance of classifier weights across training datasets, applying their method to reduce the learning of spurious correlations in Visual Question Answering (VQA) tasks. Ahuja et al. [1] analyzed IRM from a game-theoretic perspective to develop an alternative algorithm. Krueger et al. [55] proposed an approximation to the IRM problem by reducing the variance of error averages across domains. Bouvier et al. [15] tackled the same problem as IRM by re-weighting data samples.

## 2.2 Parameter Sharing Techniques

Parameter sharing is a technique in machine learning where the same parameters are used across different parts of a model. This can be done to reduce the number of parameters in a model, which can make it easier to train and generalize. Parameter sharing can be used in a variety of machine learning models, including neural networks, support vector machines, and decision trees. In neural networks, parameter sharing is often used in the convolutional layers, where the same weights are applied to different parts of the input image. This can help to reduce the number of parameters in the model, while still preserving the ability to learn complex features.

Parameter sharing can also be used in the context of domain generalization. One way to address domain generalization is to use parameter sharing to encourage the model to learn features that are invariant to changes in the domain. For example, if a model is trained on data from two different domains, one with images of cats and one with images of dogs, then parameter sharing can be used to encourage the model to learn features that are common to both cats and dogs. This can help the model to generalize to new domains, such as a domain with images of both cats and dogs.

In the context of previous works on parameter sharing techniques for improving domain generalization, Blanchard et al. [12] construct classifiers $f(x_d, \mu_d)$, where $\mu_d$ is a kernel mean embedding [71] that summarizes the dataset related to the example $x_d$. As the distributional identity of test instances is unknown, these embeddings are approximated using single test examples during testing. Theoretical results on this family of algorithms can be found in [11, 22]. Khosla et al. [50] learn a max-margin linear classifier $w_d = w + \Delta d$ for each domain $d$, from which they extract the final, invariant predictor $w$. Ghifary et al. [33] employ a multitask autoencoder to learn invariances across domains, assuming that each training dataset contains the same examples, such as photographs of the same objects under varying views.

Mancini et al. [66] train a deep neural network with a dedicated set of batch-normalization layers [48] for each training dataset. A softmax domain classifier is then used to predict the linear combination of the batch-normalization layers during testing. Similarly, Mancini et al. [65] learn a softmax domain classifier to combine domain-specific predictors at test time linearly. DInnocente and Caputo [24] investigate more sophisticated methods of aggregating domain-specific predictors.

Li et al. [57] extend Khosla et al. [50] to deep neural networks by augmenting each of their parameter tensors with an additional dimension, indexed by the training domains, and set to a neutral value

for domain-agnostic test example prediction. Ding and Fu [23] apply parameter-tying and low-rank reconstruction losses to learn a predictor that relies on shared knowledge across training domains. Hu et al. [44] and Sagawa et al. [82] weigh the importance of training distribution mini-batches proportional to their error.

## 2.3 Meta-Learning Techniques

Meta-learning is a technique that learns to learn. This means that meta-learning algorithms can learn to improve their performance on a new task, even if they have never seen that task before. Meta-learning algorithms typically do this by storing information about previous tasks and using that information to improve their performance on new tasks. This information can include the parameters of previous models, the loss functions used to train previous models, or the features that were found to be important in previous tasks.

Meta-learning has been used for a variety of tasks, including domain generalization. One way to address domain generalization with meta-learning is to use meta-learning to learn a good initialization for a model. This means that the meta-learning algorithm can learn to initialize a model in a way that makes it more likely to generalize to new domains. Another way to use meta-learning for domain generalization is to use meta-learning to learn a good optimization algorithm. This means that the meta-learning algorithm can learn to optimize a model in a way that makes it more likely to generalize to new domains.

In the context of previous works on meta-learning techniques for improving domain generalization, Li et al. [58] utilizes Model-Agnostic Meta-Learning (MAML) [27] to create a predictor that learns to adapt between training domains quickly. Dou et al. [25] employ a similar MAML strategy, along with two regularizers that encourage features from different domains to adhere to inter-class relationships and be compactly clustered by class labels. Li et al. [63] extend the MAML meta-learning approach to domain generalization instances where the categories differ across domains. Balaji et al. [7] use MAML to meta-learn a regularizer that encourages a model trained on one domain to perform well on another domain.

## 2.4 Data Augmentation Approaches

Data augmentation is a technique that artificially increases the size of a dataset by creating new data points from existing data points. This can be done by applying a variety of transformations to the data, such as flipping, rotating, cropping, or translating the data. Data augmentation is a powerful technique that can be used to improve the performance of machine learning models. This is because data augmentation can help to reduce overfitting, which is a problem that occurs when a model learns the training data too well and is unable to generalize to new data.

We already explored some of the data augmentation techniques in 1.2.3. These data augmentation has proven to be an effective strategy for addressing domain generalization [105]. Unfortunately, designing efficient data augmentation routines depends on the data type and requires significant effort from human experts. Xu et al. [99], Yan et al. [100], and Wang et al. [98] employ mixup [104] to blend examples from different training distributions. Carlucci et al. [16] create an auxiliary classification task aimed at solving jigsaw puzzles of image patches, which learns features that improve domain generalization through self-supervised learning. Albuquerque et al. [4] introduce a self-supervised task of predicting responses to Gabor filter banks to learn more transferable features.

Wang et al. [94] remove textural information from images to enhance domain generalization. Volpi et al. [92] demonstrate that training with adversarial data augmentation on a single domain can improve domain generalization. Nam et al. [72] and Asadi et al. [6] promote data representations disregarding texture and focusing on shape. Rahman et al. [79], Zhou et al. [108], and Carlucci et al. [16] offer three alternatives that utilize GANs to augment the data available during training time.

## 2.5 Conclusion

In summary, this chapter has provided an in-depth review of the related literature on domain generalization, covering a wide range of approaches that aim to address the challenges associated with learning invariant features, sharing parameters, utilizing meta-learning techniques, and applying data augmentation. The diverse range of techniques and methodologies discussed in this chapter highlight the complexity and importance of domain generalization in the broader context of machine learning and artificial intelligence.

*Chapter 3*

# Cross-Domain Class-Contrastive Learning: Finding Lower Dimensional Representations for Improved Domain Generalization

## 3.1 Introduction

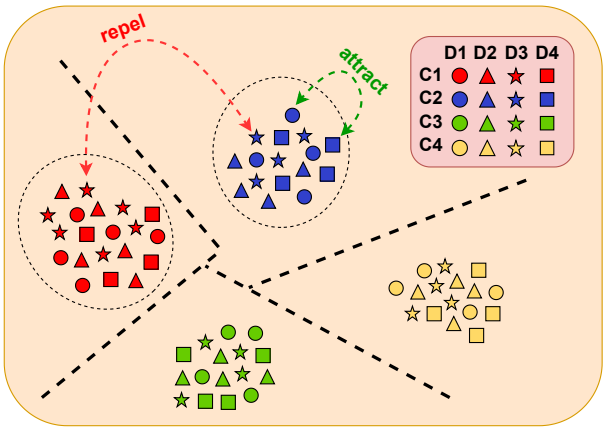



**Figure 3.1** This figure shows how Cross-Domain Class-Contrastive Learning clusters samples from different domains and classes. The color of a sample indicates its class, and the shape of a sample indicates its domain. See the legend in the top-right corner for more details. Note that the number of domains may or may not be equal to the number of classes.

The relevance of evaluating deep neural networks on test data sampled from outside of the train distribution is multi-faceted. It not only aids model deployment across varied environments but also helps reveal the learning mechanism of networks. Domain generalization (DG) is a formalized setting for Out of Distribution (OOD) generalization where a model is trained on multiple domains and tested on the same classes of an unseen domain. The ability to give good DG performance is desirable when the test conditions are either not always predictable or are difficult to gather and annotate (e.g. in self driving cars).

DG methods aim at learning domain agnostic discriminative features and not fitting domain specific ones. Some of the popular methods in DG include learning features by reversing gradients from domain labels [29], learning a common feature representation space for different domains [70], learning both domain-specific and domain agnostic features together [50], and inhibiting features corresponding to the highest gradients [46].

However, recent work [36] shows that the intuitive methods tailored for DG, fail to give consistent improvement over a simple Empirical Risk Minimization (ERM) when trained with a similar strategy across the different datasets. Sivaprasad *et al.* [85] explains this observation arguing that the same phenomenon that leads to shortcut learning in IID (fitting the low lying variance), improves performance in OOD, specifically DG. Figure 3.2 shows how the probability of learning $R_{ABC}$ is higher when network learns the least number of features to explain the train data ($R_{BC}$), compared to learning $R_B$ or $R_C$ independently. In the presence of multiple domains ($B$ and $C$), the network has a higher probability of learning domain agnostic features $R_{ABC}$ compared to learning from individual domains.

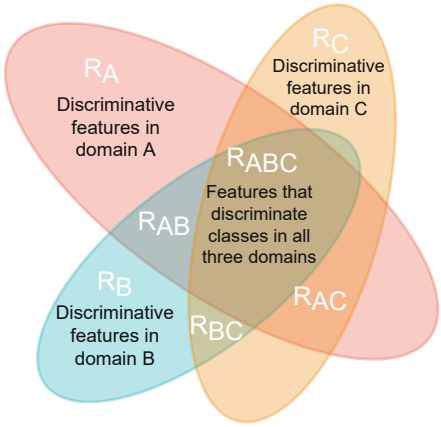



**Figure 3.2** B, C are the training domains, and A is the test domain. $R_i$ are the discriminative features (in parameter space) of domain $i$. $R_{BC}$ is an intersection of features of $R_B$ and $R_C$, which is the smallest set of variances that explains the class separation in train data. Also, $\frac{|R_{ABC}|}{|R_B|} \leq \frac{|R_{ABC}|}{|R_{BC}|}$ and $\frac{|R_{ABC}|}{|R_C|} \leq \frac{|R_{ABC}|}{|R_{BC}|}$, (here, $|S|$ represents the cardinality of set $S$) making a case that a model that learn **compact representation** on multiple domains has higher chance of learning **domain agnostic features**.

We hypothesize, that learning a more compact/low dimensional representation ($R_{BC}$) is beneficial to DG. To this end, we perform experiments to bring out the correlation between the performance of a model in DG with the dimensionality of the feature space. The correlation holds true across six different popular image classification architectures studied in our work. We measure the dimensionality of feature space by using the cardinality of the dominant singular values of the weight matrix that projects data to

the last layer embedding. We show how this measurement guarantees an upper bound to dimension of the learned feature space.

We take this idea forward and propose a new training strategy to learn domain agnostic feature representation for DG. We propose Cross-Domain Class-Contrastive (CDCC) loss, a modified version of Supervised Contrastive Loss (SCL) [51] and similar to the Cross-domain Contrastive loss proposed by [97] for domain adaptation. CDCC brings samples of the same class from all the domains together and pushes the samples that belong to different classes (irrespective of its domain) farther apart, resulting in better domain agnostic learning. Figure 3.1 shows how Cross-Domain Class-Contrastive (CDCC) Loss applied on batches sampled from different domains explicitly learns domain agnostic features. Features from the same class fall in the same cluster irrespective of their domain. Over the epochs, the features become clustered such that features of same class from different domain are 'attracted' to form a cluster, and features from different classes in the same domain are pulled apart or 'repelled' into separate clusters. The embedding space post-training is clustered such that all elements from the same class fall in the same cluster irrespective of their domain.

We establish SOTA results on five popular DG benchmarks using this method. We show that using CDCC reduces the dimensionality of the feature space tying together this finding with the previous claim. Overall, our work makes the following contributions:

- We show that the DG performance of models are correlated to the implicit dimensionality of its learned representation.
- We propose a new loss function (CDCC) that beats state-of-the-art results on five different DG benchmarks.
- We show that the proposed loss function helps reduce the dimensionality of the learned representation resulting in better DG performance.

## 3.2 Related Work

Our discussions are limited to efforts exploring DG properties of deep neural networks on the image classification tasks. We refer the reader to the work by Moreno *et al.* [68] for a more comprehensive discussion on DG, discussing literature beyond the task of image classification, including methods from the pre-deep learning era.

Some popular image classification benchmarks for DG evaluation are: PACS [57], VLCS [26], Office-home [91], RMNIST [33], Domain-Net [74] and CMNIST [5]. The statistics of these datasets are given in Table 3.1

Efforts in DG in neural networks can be broadly categorized into three categories [95]. Firstly, significant efforts have been made in improving domain generalization through data generation and augmentation [103, 83, 77]. These techniques improve the variances in the training data aiming to

| Dataset | #D | Domains | #C | #Images |
|---|---|---|---|---|
| RMNIST [33] | 6 | 0°, 15°, 30°, 45°, 60°, 75° | 10 | 70000 |
| CMNIST [5] | 2 | Red, Green | 2 | 120000 |
| DomainNet [74] | 6 | Clipart, Infograph, Painting, Quickdraw, Real, Sketch | 345 | 586575 |
| PACS [57] | 4 | Photo, Art-Painting, Cartoon, Sketch | 7 | 9991 |
| VLCS [26] | 4 | Caltech101, LabelMe, SUN09, VOC2007 | 5 | 10729 |
| Office-Home [91] | 4 | Art, Clipart, Product, Photo | 65 | 15558 |

**Table 3.1** This table presents the statistics of popular datasets used for evaluating DG performance. **#D**, **#C** and **#Images** represent the number of domains, number of classes and total number of images in the dataset respectively.

improving the generalization of the model. Data augmentation is done by shape and style manipulation [103], adversarial augmentation [83] and image generation [77]. These methods move towards domain randomization where, the target domain is another domain to the network. Methods like data augmentation [14, 109, 99] has shown to consistently improve DG in deep networks. It is worth noting that data augmentation is known to enhance the robustness of feature learning towards different kinds of noise and adversarial attacks [41, 40]. These ideas have shown to have merits and data augmentation is a part of all DG training protocols.

The second major direction in tackling DG has been through learning strategies. Gradient manipulation is one of the primary efforts in this direction [29, 46]. RSC [46] iteratively masks the features corresponding to the highest gradient in every training step. The hypothesis behind RSC formulation follows that feature that most contribute to the prediction when masked; the neural network learns alternate features that are more domain agnostic. Gradient reversal [29] aims to remove domain specific features by reversing the gradients from an additional domain prediction branch.

[29] involves branching the network at a selected layer, and using a separate classification head to predict the class label and the domain label. The gradient from the domain prediction head is reversed such that the feature at the branching layer is devoid of domain information. Both these methods implicitly do feature learning to remove domain information from features and improve class discrimination. More recent methods like IDFM [85] also achieve performance in OOD through feature learning.

The third category of methods involve representation learning. Most of these methods employ strategies that implicitly improve feature learning. The popular attempts here includes ensemble learning [65], and feature disentanglement [50, 76]. Efforts have been made to understand the effect of spectral norm regularization for improving generalization performance [102]. It involves penalizing the high spectral

norm of weight matrices of the neural network. It results in better generalizability for sensitivity to input perturbations.

[95, 107] gives a comprehensive list of recent works that tackle DG. Recently, Gulrajani and Lopez-paz [36] propose a test bench that facilitates a fair comparison of methods by freezing training strategies like augmentation across various benchmarks. They show that the DG tailored methods fail to improve over the ERM baseline. Sivaprasad *et al.* [85] attribute this success of ERM to the fact that with more domains in the training set, the neural network already learns domain agnostic features (Figure 3.2). Comparing with an alternate OOD framework, they show how in DG framework, neural networks have incentive for learning domain-agnostic features.

We take this idea forward and hypothesize that the low lying variance that enables performance in DG (with multiple domains in train) is also the lowest dimensional representation. We show empirical evidence towards this hypothesis by comparing the rank of the weight matrix that projects data into the feature space. We propose a supervised Cross-Domain Class-Contrastive feature learning method and show that it outperforms the current SOTA methods. We empirically show that the upper bound of the span of the feature space learnt through this method is lower than the prior art.

## 3.3 Method

In this section, we propose a method to measure the upper bound of the dimensionality of learnt feature space. This measure is later shown to correlate with the generalization performance of the models. We also describe the proposed Cross-Domain Class-Contrastive loss function.

### 3.3.1 Computing the dimensionality

For a given task, a robust representation of data is believed to be found at the last layer of the neural network. This assumption is widely used in transfer learning applications. We call this representation $H_i$ and compute the dimension of this feature space (minimum number of independent features that are necessary and sufficient for classification).

It is to be noted that, the rank of the matrix of train features alone cannot give an estimate of dimension of $H_i$ as it gives no indication to the the rank of feature matrix of OOD test samples. We use the rank of the weight matrix ($W$) that projects the previous features ($H_{i-1}$) to $H_i$ to give an upper bound to the dimension of $H_i$ irrespective of the input data. That is, given $H_i = W \times H_{i-1}$ the dimension of the span of $H_i \leq \textit{rank}(W)$. Hence, we argue that the rank of $W$ is a hard constraint on the dimension of the span of $H_i$.

To get a measure of the dimension of learnt feature space, we use the minimum number of singular values required to reconstruct $W$ such that the training accuracy does not reduce below a threshold $h$. This proxy can give a useful upper bound of the dimension of the learnt feature space. That is,

features before this layer already occupy a subspace and the resulting features after the layer have its dimensionality at most as the rank of the weight matrix.

Post SVD decomposition, we get the singular value matrix $S$ as a diagonal matrix. We use the top $n$ values of this matrix such that the reconstruction of $W$ with the $n$ values does not drop the neural network's performance by more than $h$. $h$ is a hyper-parameter, and we choose it to be 0.5% across all our experiments. Since the train performance increases monotonically with the value of $n$, we use binary search to faster determine $n$ for each run (note that we use only the training data for this purpose). Simply choosing the knee of the singular value spectrum gives inconsistent results as some of the singular values might correspond to spurious variances. The proposed method ensures that we consider the discriminative variances in the train data, and we call this the effective rank of the weight matrix.

### 3.3.2 Cross-Domain Class-Contrastive Learning

Traditional DG formulation follows that we train on $N$ domains and test on the $(N+1)^{th}$ domain. We propose Cross-Domain Class-Contrastive (CDCC) loss for this setting. The network consists of two branches: a feature learning branch $B_{CDCC}$ and a classifier learning branch $B_{CL}$.

The feature learning branch learns a better representation with CDCC loss, and the classifier branch uses cross-entropy loss for classification learning. The feature learning head and the classification head are jointly trained. The objective for minimization in DG setting with Hybrid loss consisting of CDCC loss, and cross-entropy loss is as follows:

$$\mathcal{L}_{Hybrid} = \alpha \cdot \mathcal{L}_{CDCC} + (1-\alpha) \cdot \mathcal{L}_{CE} \quad ; \alpha \in [0,1] \tag{3.1}$$

where $\mathcal{L}_{CDCC}$ is the Cross-Domain Class-Contrastive Loss, $\mathcal{L}_{CE}$ is the Cross Entropy Loss, and $\alpha$ is the weight factor that takes a weighted average of both the losses. For the hybrid model, $\alpha$ linearly decreases from 1 to 0 over the epochs. That is, $\alpha = 1$ when $epoch = 0$ and $\alpha = 0$ when $epoch = epoch_{max}$. $\alpha$ determines the trade-off between the classification and representation losses. The linear decrease in $\alpha$ results in the model to emphasize more on learning a better representation during the initial epochs and then gradually shifting the emphasis to learning a better classifier till the finishing epoch. Figure 3.3 shows a visual of how the hybrid model works. In this figure, we consider 4 input images as *House-Cartoon*, *Giraffe-Photo*, *Dog-Cartoon* and *Giraffe-Art Painting*. The (green colored) contrastive learning branch on the right brings closer the representations of *Giraffe-Photo* and *Giraffe-Art Painting* (different domains, same class), and brings apart the representations of *House-Cartoon* and *Dog-Cartoon* (different class, same domain) in the feature space. The (blue colored) classifier learning branch learns a classifier for the input samples.

Unlike vanilla contrastive learning formulation, which creates tuples consisting of (input sample, class label) pairs, CDCC uses triplets of the form (input sample, class label, domain label). Supervised contrastive formulations like CDCC follows that data points with the same class are positives and different class are negatives irrespective of the domain. We sample batches such that for each anchor point,

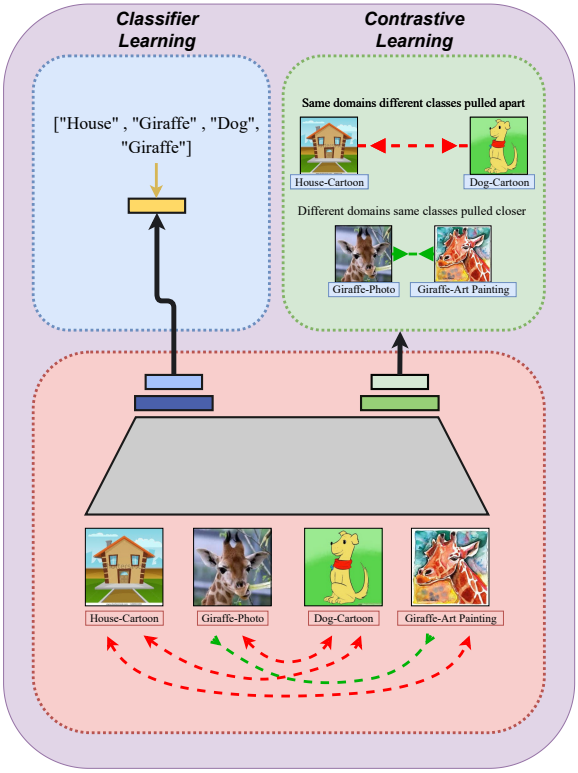



**Figure 3.3** This figure shows how the classifier learning and contrastive learning branches of the hybrid model contribute to learning a classifier over an improved representation in the feature space. The green colored bi-directional arrows show the 'attraction' among the representations of the samples. Similarly, the red colored bi-directional arrows show the 'repulsion' among the representations of the samples.

atleast 80% of the negative samples are picked from the same domain but different class and atleast 80% of the positive samples are picked from the same class but different domain (refer Figure 3.4).

Considering one such mini-batch with $N$ datapoints, we have $N$ triplets of the form (input sample, class label, domain label), $(x_k, y_k, d_k)_{k=1...N}$. In each input batch of the model, we augment the input to create a pair of two distorted images with the same class and domain labels. We use only the augmented pair (not the actual samples) making the batch size $2N$. The effective dataset for training hence comprises of $2N$ triplets: $(\tilde{x}_l, \tilde{y}_l, \tilde{d}_l)_{l=1...2N}$, where $\tilde{x}_{2k}$ and $\tilde{x}_{2k-1}$ are two random augmentations of $x_k$ $(k = 1...N)$ and $(\tilde{y}_{2k-1}, \tilde{d}_{2k-1}) = (\tilde{y}_{2k}, \tilde{d}_{2k}) = (\tilde{y}_k, \tilde{d}_k)$. Using these conventions, the loss $\mathcal{L}_{CDCC}$ is given as follows,

$$\mathcal{L}_{CDCC} = \sum_{i \in \{1...2N\}} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \bigodot z_p/\tau)}{\sum_{a \in A(i)} \exp(z_i \bigodot z_a/\tau)} \tag{3.2}$$

Where, the $\bigodot$ denotes the inner (dot) product, $\tau \in \mathcal{R}^+$ is a scalar temperature parameter, and for $i \in \{1...2N\}$:

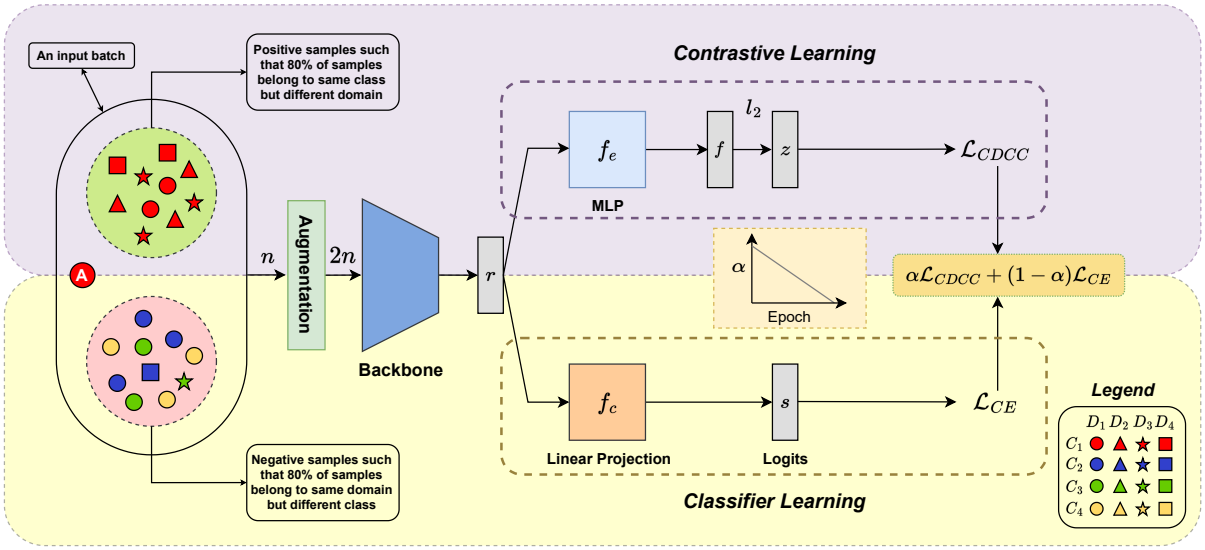



**Figure 3.4** The above figure shows the hybrid architecture used for CDCC learning.

- $z_i$ is the Normalized Embedding Vector of $\tilde{x}_i$.
  That is $z_i = B_{CDCC}(x_i) \ / \ ||B_{CDCC}(x_i)||$
- $A(i) \equiv \{1...2N\}$ excluding the $i^{th}$ sample
- $P(i) \equiv \{p \in A(i) | \tilde{y}_p = \tilde{y}_i\}$

Here, $P(i)$ is the set of all samples belonging to the same label as that of the $i^{th}$ sample except for the $i^{th}$ sample itself. In other words, $P(i)$ is the set of all *positives* of the $i^{th}$ sample. Stratified sampling of $P(i)$ based only on the domains (sampling from other domains) helps the contrastive loss to bring samples from different domains with the same label closer to each other and pull samples with different labels apart from each other (Figure 3.1). This results in explicitly learning the domain agnostic features. It is to be noted that this is a modified version of NT-Xent Loss as proposed in [17], and is not to be misinterpreted as an unsupervised triplet loss formulation.

Figure 3.4 shows the end-to-end architecture of our hybrid model. It demonstrates a case where the input samples consist of 4 domains (represented by different shapes) and 4 classes (represented by different colors). Refer the legend provided at the bottom-right corner of the figure to understand the convention for the same. For the input batch, we consider an anchor $A$ belonging to domain $D_1$ and class $C_1$. For this anchor, we sample equal number of positive and negative examples as shown in the figure. Positive samples are randomly sampled (with repetition) such that 80% of samples belong to same class but different domain. Similarly, negative samples are randomly sampled (with repetition) such that 80% of samples belong to same domain but different class. Here, $r$ is the output embedding of the backbone. This embedding is forwarded to both Contrastive Learning branch and Classifier Learning branch. The

| Training Domains | Test Domain | Number of neurons in last layer | | | | |
|---|---|---|---|---|---|---|
| | | **8** | **64** | **256** | **512** | **1024** |
| Art Painting, Cartoon, Sketch | Photo | 89.10 | 90.48 | 88.38 | 89.10 | 88.50 |
| Cartoon, Photo, Sketch | Art Painting | 73.19 | 73.54 | 71.58 | 72.12 | 70.85 |
| Art Painting, Photo, Sketch | Cartoon | 77.47 | 75.90 | 76.62 | 75.85 | 77.22 |
| Art Painting, Cartoon, Photo | Sketch | 82.01 | 80.25 | 78.29 | 78.47 | 78.72 |
| **Average** | | 80.44 | 80.04 | 78.72 | 78.89 | 78.82 |

**Table 3.2** This table lists the DG performance achieved on PACS dataset using ResNet-18 backbone with varying number of neurons in the last fully connected layer. An SGD optimiser with a learning rate of 0.01, and a batch size of 32 was used for this experiment.

Contrastive Learning branch consists of $f_e$ : A multi-layer perceptron, $f$ : The output embedding of $f_e$, and $z$ : An embedding obtained after performing $l_2$ normalization on $f$. The Classifier Learning branch consists of $f_c$ : A Linear Projection (or a single-layer perceptron) and $s$ : The logits obtained as the output of $f_c$. Contrastive loss ($\mathcal{L}_{CDCC}$) is computed over the output of the Contrastive Learning branch $(z)$ and Cross-Entropy loss ($\mathcal{L}_{CE}$) is computed of the output of the Classifier Learning branch $(s)$. Lastly, a hybrid loss with a weight factor of $\alpha$ is computed over $\mathcal{L}_{CDCC}$ and $\mathcal{L}_{CE}$ as shown.

## 3.4 Experiments and Results

In this section, we detail our four experiments. In the first experiment we show the motivation to why the dimention of the network can be correlated to the generalization performance in DG. We systematically reduce the number of neurons in the last fully connected layer of the neural network and report its generalisation performance. In the second experiment, we compute the dimensionalities across different backbones and try to understand the correlation between the dimensionality and domain generalisation performance. For the third experiment, we compute the dimensionalities across the backbones using CDCC loss further showing that reduced dimensionality improve domain generalisation performance. For the last experiment, we pick the best performing backbone from the third experiment and achieve state-of-the-art performance on 5 popular DG benchmarks.

### 3.4.1 Improving generalisation by hard constraining low dimensionality

In this experiment, we use PACS [57]. We train a ResNet-18 backbone (pre-trained on ImageNet dataset) with an SGD optimizer with a learning rate of 0.01 and a batch size of 32.

| | PACS | | VLCS | |
|---|---|---|---|---|
| | w/o | w/ | w/o | w/ |
| InceptionV2 | 5.18 | 4.38 | 3.14 | 3.01 |
| ResNet50 | 5.89 | 5.07 | 3.89 | 3.39 |
| DenseNet121 | 5.95 | 5.32 | 3.92 | 3.22 |
| ResNet18 | 5.95 | 5.23 | 3.90 | 3.33 |
| VGG | 8.35 | 7.69 | 4.34 | 3.61 |
| AlexNet | 13.98 | 11.11 | 25.21 | 18.99 |

**Table 3.3** This table lists the upper bounds over the dimensionality of the feature space learned by different backbones. The dimensionalities of the backbones with (w/) and without (w/o) CDCC loss are shown in the two columns for both PACS and VLCS datasets.

We created 5 different models by modifying the number of neurons in the last fully connected layer of the ResNet-18 backbone to 8, 64, 256, 512, and 1024 respectively. By doing this, we constrain the dimensionality of the weight matrix corresponding to the last fully connected layer. For each of these models, we train the model 4 times, each time considering a different domain as a test domain and the remaining domains as training domains.

The results of these experiments are reported in Table 3.2. We observe that the average performance is maximum for the model with minimum number of neurons in the last fully connected layer. Furthermore, with the increasing number of neurons in the last layer, the generalization performance came down, albeit with an increased number of parameters.

### 3.4.2 Computing dimensionalities across different backbones

We use the two DG benchmark datasets, namely PACS [57] and VLCS [26] for this experiment. In this experiment, we measure the dimensionality of the learnt feature space using the rank of weight matrix as explained in Sec. 3.3.1. We compute the dimensionality on 6 different backbones, InceptionV2 [88], ResNet50 [38], DenseNet121 [45], Resnet18 [37], VGG [84] and AlexNet [54]. For a fair comparison across these backbones, we modify the last layers to ensure that the last weight matrix is always a square matrix of size $512 \times 512$.

For training the networks, we use a learning rate of 0.01 for training PACS (except for AlexNet, which uses a learning rate of 0.001) and 0.001 for VLCS. We base our choice of training strategies, optimizers, and augmentations on DomainBed [36] with ImageNet pre-trained weights for all the backbones. We report the average effective rank of the last layer matrix over five runs.

In Table 3.3, the w/o columns corresponding to the PACS and VLCS dataset shows the upper bound of the dimensionality of the feature space learnt by each backbone. We observe that the performance of a backbone is inversely correlated to its corresponding dimensionality. The yellow line in Figure 3.5 shows this inverse correlation of the model performance with its dimensionality. This provides evidence to the hypothesis that a model that learns a low dimensional feature representation performs better in DG setting.

### 3.4.3 Dimensionality with CDCC

In the third experiment, we train our five different backbones with CDCC loss. All training hyperparameters are the same as the above experiment (3.4.2) except for having the extra loss function (CDCC loss). Table 3.3 shows that, when trained with CDCC loss, the model consistently achieves a lower dimensionality across all backbones when compared to the counterpart model trained without CDCC. Also, In Figure 3.5, the red colored plot (w/ CDCC) shows that we always achieve a better performance when a backbone is trained with CDCC loss, compared to when trained without CDCC loss. This further strengthens the hypothesis that a model's performance is inversely correlated to its dimensionality of the feature space.

These observations suggest that contrastive loss functions specific to DG, (like CDCC) can further reduce the dimensionality of the learnt feature space and improve the DG performance. Our experiments empirically suggest that constraining feature dimensionality could be a promising direction for DG.

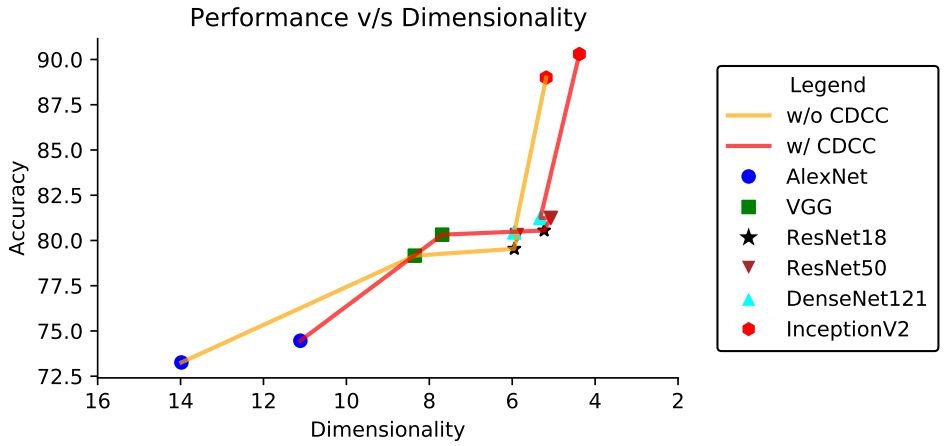



**Figure 3.5** This figure shows the correlation between the DG performance (Y-Axis) and the dimensionality (X-Axis) of the learnt feature space for PACS dataset. Please note that the X-axis is flipped here.

| | PACS | VLCS | Office-home | CMNIST | RMNIST | Average |
|---|---|---|---|---|---|---|
| GRL | 83.69 | 77.38 | 70.20 | 50.50 | 98.49 | 76.05 |
| IRM | 82.90 | 77.20 | 66.70 | 59.16 | 97.70 | 76.73 |
| DANN | 84.00 | 77.70 | 65.50 | 73.03 | 89.10 | 77.87 |
| C-DANN | 81.70 | 74.00 | 64.70 | 73.03 | 96.30 | 77.94 |
| RSC | 84.77 | 78.80 | 70.80 | 61.20 | 98.23 | 78.76 |
| MLDG | 82.40 | 77.10 | 67.60 | 71.64 | 98.00 | 79.35 |
| MMD | 82.80 | 76.70 | 67.10 | 73.35 | 98.10 | 79.61 |
| DRO | 83.10 | 77.50 | 67.10 | 73.35 | 97.90 | 79.79 |
| CORAL | 83.60 | 77.00 | 68.60 | 73.35 | 98.10 | 80.13 |
| Mixup | 83.70 | 78.60 | 68.20 | 73.34 | 98.10 | 80.38 |
| ERM | 89.04 | 78.84 | 71.95 | 74.35 | 99.20 | 82.67 |
| CDCC | **90.30** | **79.16** | **72.07** | **74.46** | **99.65** | **83.12** |

**Table 3.4** This table compares the performance of InceptionV2 with CDCC loss, with other popular algorithms from DomainBed. The algorithms are sorted by their average performance across the five datasets.

### 3.4.4 DG benchmarks with CDCC

For this experiment, we choose the best performing backbone from the previous experiment (3.4.3) i.e. InceptionV2 with CDCC Loss. To find evidence for our claim that explicit feature learning gives improved results in a DG setting (as it seems from Figure 3.5), we train CDCC with InceptionV2 backbone on 5 different DG datasets: PACS [57], VLCS [26], Office-home [91], RMNIST [33], and CMNIST [5].

We use the hyperparameters, and the random-seed, as used in DomainBed [36] to maintain uniformity of comparison. We use a learning rate of 0.01 for training PACS, RMNIST, and CMNIST, and a learning rate of 0.001 for training VLCS and OfficeHome datasets. We use a batch size of 32 per GPU and a total of four GPUs. We use image augmentations as in DomainBed, which follows that MNIST datasets, namely CMNIST [5] and RMNIST [33] are not augmented. To create augmented pairs for CDCC, we use Random-Resized-Crop, Random-Horizontal-Flip, and Color-Jitter augmentations (also used in DomainBed). Our preprocessing is also similar to [36]. We compare our results against popular methods, namely: C-DANN [62], IRM [5], MLDG [58], DRO [82], MMD [60], ERM [36], CORAL [87], Mixup [99], RSC [46], DANN [30] and GRL [29].

It is to be noted that the number of epochs is a significant hyper-parameter for this experiment as it uses the hybrid loss. Loss weighting ($\alpha$) at every step is a function of the current epoch number Sec. 3.3.2. Therefore, unlike simple categorical cross-entropy loss used in the prior art, the number of epochs is a vital modeling choice which we maintain as 150 across all runs. Despite the significance of the number of epochs, we employ validation based model selection as explained by [36].

We run the CDCC model $D$ times for each of the five datasets, where $D$ corresponds to the number of domains in each dataset. In each run, the accuracy is computed on the kept out domain. The accuracy on a dataset is the average of $D$ runs. This is repeated five times for each dataset, and the average is reported in Table 3.4. The results clearly show the consistent improvement of CDCC over the other methods.

CDCC improves over the best DG specific model on the PACS dataset by 5.5% and above 1% over the ERM. Given that none of the DG specific methods improved over ERM on PACS, this statistically significant improvement shows the efficacy of CDCC in DG. We get an average gain of roughly 3% when compared against the next best DG specific method across the five datasets. The evidence suggest that feature learning can improve DG. By establishing state-of-the-art results on the benchmarks, we suggest that feature learning is a promising direction for improving DG going forward.

## 3.5 Conclusion

Domain generalization is a much-coveted property for neural networks. Various datasets and methods have been proposed over the years to evaluate and improve the performance of neural networks in DG. Prior art focused on learning better feature representations that are robust across domains. Recent explorations showed that neural networks already learn domain agnostic features with multiple domains in the train data, outperforming implicit methods of learning domain agnostic features. We show that using explicit supervised feature learning improves DG. We propose a modified Supervised Contrastive Learning method called Cross-Domain Class-Contrastive loss and show consistent improvement over ERM. As the second key takeaway, we show how the dimension of the feature space learned can explain the generalization performance of networks in DG. We show this by comparing the dimensionality of features across backbones, both with and without the proposed loss. Across all the studies, we observe that a lower dimensional feature representation leads to better performance.

*Chapter 4*

# Going beyond Traditional Domain Generalization

In the traditional setting of domain generalization, it is assumed that enough samples are available for all classes across all training domains. However, this assumption is often not met in real-world scenarios where some classes may be missing, or there may be an uneven distribution of samples across classes and domains, leading to data imbalance. This issue is especially relevant as many real-world distributions tend to follow a long-tailed pattern. To investigate the impact of data imbalance, we conduct a set of experiments on the PACS dataset. Later in this chapter, we also discuss some additional findings that were discovered during the experimentation on the PACS dataset.

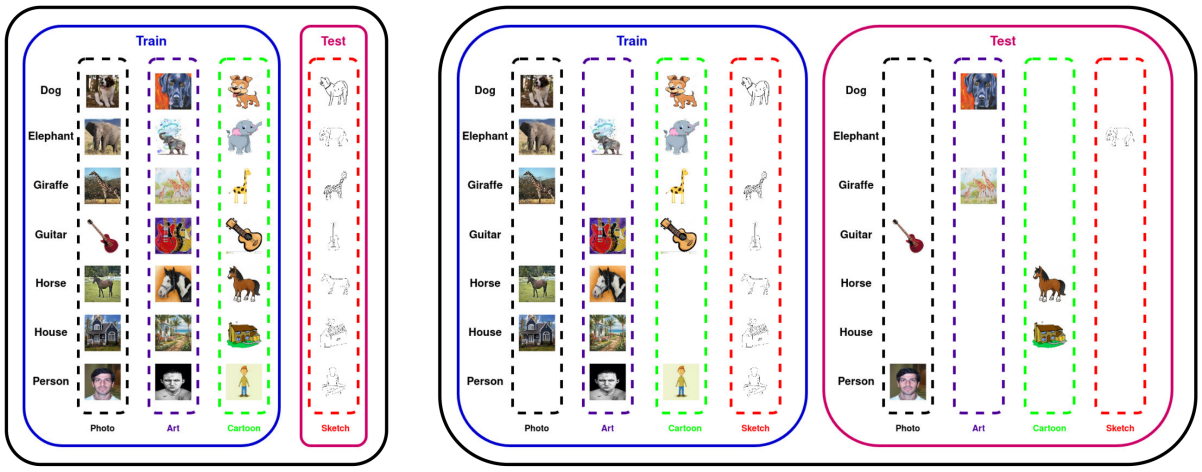
## 4.1 Class Wise Domain Generalization

We start this chapter with an experiment in the Class-Wise-Domain-Generalization framework proposed by Sarath et al. [85]. Within this setting, we can access all domains during training, though not necessarily every class from each domain. For each class, all examples from a single domain (randomly selected) are reserved for the test set, while the remaining samples from other domains are designated for the training set. This creates a more realistic and applicable environment for investigation. However, due to the model's inclination to learn simpler discriminative features, the lack of access to all classes across domains heightens the likelihood of misclassification. To illustrate this, imagine a scenario utilizing the PACS dataset, where all training samples are from the Art-Painting domain, while test samples belong to the Sketch domain. Suppose we substitute Giraffe class samples in the training set with Giraffe samples from the Sketch domain. In that case, the model is inclined to learn basic sketch image features and classify them as giraffe. Consequently, even non-giraffe test samples may be misclassified as giraffe during inference, as they share sketch features. In this configuration, using Empirical Risk Minimization (ERM) for training results in a reduced performance during inference compared to the traditional DG setting, even though the test domain is partially available during training, unlike the traditional DG setting.

We run an ERM with Inception-ResNetV2 backbone in CWDG setting for 50 epochs. For classes Dog, Elephant, Giraffe, Guitar, Horse, House, and Person, we keep test domains as Art-Painting, Sketch,

Art-Painting, Photo, Cartoon, Cartoon, and Photo, respectively (Please see Table 4.1). We achieved an average accuracy of **82.64** on five runs of this experiment. Note that this accuracy is much lesser compared to **89.11** by ERMs in traditional DG setting.

Throughout this chapter, for all experiments (except those in section 4.3), it is assumed that models are pre-trained on ImageNet and that the Stochastic Gradient Descent (SGD) optimizer is utilized unless mentioned otherwise.




(a) TDG (b) CWDG

**Figure 4.1** This figure shows the train-test split on some sample images of the PACS dataset in (a) Traditional DG (TDG) setting as shown on the left, and b. Class-Wise-DG (CWDG) setting as shown on the right.

## 4.2 Long-Tailed Domain Generalization

In this scenario, we assume that the training samples adhere to a long-tailed distribution, which is common in situations involving data imbalances. We consider two different cases for Long-Tailed distribution in DG:

- **ClassLT**: In this setting, the distribution of samples across classes follows a long-tailed distribution, while the total number of samples across all domains is kept relatively equal. This results in some classes having more influence during training compared to others.
- **DomainLT**: In this setting, the distribution of samples across domains follows a long-tailed distribution, while the total number of samples across all classes is kept relatively equal. This results in some domains having more influence during training compared to others.

It is important to note that in all of these scenarios, the quantity of samples in the test set (those from the target domain) is maintained equal to the number of samples from that target domain in the initial dataset (and not from the ClassLT or DomainLT variants of it).

When dealing with a distribution that exhibits a long tail, we use the term **common ratio** to describe the ratio between two consecutive values in the distribution. Note that this common ratio always stays between zero and one, i.e., $0 < common\,ratio \leq 1$. For example, if we have a long-tailed distribution of samples across classes such as 100, 80, 64, 51, 41, and 33, we can determine that this is a long-tailed distribution with a common ratio of roughly 0.8. This is because the ratio between each value and the one before is approximately 0.8 in this specific instance. In the long-tailed setting, the order of the classes (or domains) can result in certain classes (or domains) having more influence than others. For instance, if we have classes A, B, and C (in that order) with a long-tailed distribution across classes as 100, 50, and 25 (i.e., a common ratio of 0.5), switching the order of the classes to B, C, and A would change the ordering and result in class B having the highest number of samples (100), class C having 50 samples, and class A having only 25 samples. As a result, during training, class A would have the least influence instead of the most influence, as it did in the previous ordering. Therefore, it is crucial to consider the impact of varying the order of domains and classes in a DG scenario and understand how it affects the performance using ERMs, showing that it is harder for ERMs to perform in a long-tailed setting compared to a traditional DG setting.


**Figure 4.2** This figure shows the distribution of samples across domains and classes in the form of a heat map. The left image depicts the original distribution of samples, middle image shows the distribution for ClassLT, and the right image shows the distribution for DomainLT. Note that the common ratio followed for ClassLT is nearly 0.68, and that for DomainLT is nearly 0.46

| Domain | Original | | ClassLT | | DomainLT | |
|---|---|---|---|---|---|---|
| | TDG | CDCC | TDG | CDCC | TDG | CDCC |
| Sketch | 85.85 | 86.31 | 66.90 | 75.99 | 78.04 | 83.57 |
| Cartoon | 85.25 | 86.48 | 81.30 | 83.70 | 81.25 | 85.71 |
| Art-Painting | 88.80 | 91.50 | 85.76 | 90.91 | 83.21 | 85.36 |
| Photo | 96.65 | 98.38 | 94.00 | 93.83 | 96.79 | 97.14 |
| Average | 89.14 | 90.67 | 81.99 | 86.11 | 84.82 | 87.95 |

**Table 4.1** This table displays the outcomes of an Empirical Risk Minimization (ERM) run on the long-tailed versions of the PACS dataset (single run) under both Traditional Domain Generalization (TDG) and Cross-Domain Class Contrastive (CDCC) settings.

### 4.2.1 Experiments showing that the performance of CDCC is more robust to Long-Tailed Distributions

We perform our experiments on the Original, ClassLT, and DomainLT variants of our PACS DG dataset (the distribution of which are as shown in Figure 4.2). We train them both using TDG (Traditional DG) and using CDCC (as described in Chapter 3) using an Inception-ResNetV2 backbone. A learning rate of 0.01 was used throughout. For a single run, we observe that there is a drastic drop in performance for both ClassLT and DomainLT in the case of ERMs. Whereas, in the case of CDCC, the performance drop is much lesser than that of the ERMs. This shows that using CDCC has a lesser effect on the performance even when trained on a long-tailed dataset, making it more robust than ERMs. Please see Table 4.1 for results.

### 4.2.2 Experiments showing that the order of performances across test domains is agnostic to the order of training domains in the long-tailed setting.

From Table 4.1, we see that the order of performances across the test domains on the original PACS dataset in traditional DG setting is in the following order:

$$Cartoon < Sketch < Art - Painting < Photo$$

The results of our experiments demonstrate that this performance hierarchy persists even when the training domains exhibit a long-tailed distribution. Furthermore, this performance ranking is maintained regardless of the variations in the arrangement of training domains within a long-tailed distribution across different domains.

To show that the order of performances across the test domains remains consistent, we first find this order of performances on a Balanced-PACS dataset. The Balanced-PACS dataset is created from the PACS dataset, such that the number of samples across all domains and across all classes remains the same. It can be thought of as a long-tailed distribution across both classes and domains, with a common ratio of 1. We keep 80 samples per class per domain for creating the Balanced-PACS dataset. This dataset ensures that there are no biases whatsoever due to data imbalances, ensuring that every domain and every class has the same influence on feature learning. In a traditional DG setting, we train an ERM on the Balanced-PACS dataset with an Inception-ResNetV2 backbone for this experiment. We use a learning rate of 0.01 across all the experiments. For all these experiments, the model was trained for 30 epochs. Table 4.2 shows that the order of performances remains the same for the Balanced-PACS dataset, i.e., $Cartoon < Sketch < Art - Painting < Photo$. All values in the table are the averages across five different runs. Note that this table shows all the per-class-per-domain accuracies to understand the influence of every class on the overall accuracy of the test set. As the dataset is balanced, the accuracy on a test domain is simply the average of all the per-class-per-domain accuracies for that test domain.

| Domain | Statistic | Dog | Elephant | Giraffe | Guitar | Horse | House | Person | **Accuracy** |
|---|---|---|---|---|---|---|---|---|---|
| **Sketch** | MIN | 31.25 | 88.75 | 62.50 | 96.25 | 91.25 | 88.75 | 73.75 | 81.61 |
| | MAX | 57.50 | 95.00 | 78.75 | 100.00 | 96.25 | 98.75 | 91.25 | 84.29 |
| | AVG | 44.50 | 92.75 | 68.50 | 98.50 | 94.50 | 95.50 | 83.75 | **82.57** |
| | STD-DEV | 10.99 | 2.40 | 6.46 | 1.63 | 2.09 | 4.01 | 6.67 | 1.04 |
| **Cartoon** | MIN | 57.50 | 61.25 | 86.25 | 97.50 | 61.25 | 95.00 | 63.75 | 81.25 |
| | MAX | 76.25 | 80.00 | 96.25 | 100.00 | 86.25 | 97.50 | 91.25 | 82.86 |
| | AVG | 67.00 | 71.50 | 92.75 | 99.25 | 70.25 | 96.00 | 79.00 | **82.25** |
| | STD-DEV | 6.65 | 6.93 | 3.79 | 1.12 | 10.05 | 1.05 | 10.77 | 0.61 |
| **Art-Painting** | MIN | 80.00 | 83.75 | 90.00 | 80.00 | 80.00 | 91.25 | 67.50 | 85.54 |
| | MAX | 91.25 | 91.25 | 98.75 | 91.25 | 88.75 | 98.75 | 82.50 | 87.50 |
| | AVG | 83.75 | 88.00 | 95.25 | 84.00 | 85.25 | 94.50 | 72.50 | **86.18** |
| | STD-DEV | 4.59 | 3.38 | 3.47 | 4.37 | 3.35 | 2.74 | 5.93 | 0.80 |
| **Photo** | MIN | 83.75 | 92.50 | 88.75 | 95.00 | 87.50 | 100.00 | 98.75 | 94.29 |
| | MAX | 96.25 | 100.00 | 98.75 | 98.75 | 96.25 | 100.00 | 100.00 | 97.32 |
| | AVG | 91.25 | 96.75 | 95.25 | 97.00 | 91.50 | 100.00 | 99.75 | **95.93** |
| | STD-DEV | 4.76 | 3.38 | 4.09 | 1.43 | 3.24 | 0.00 | 0.56 | 1.15 |

**Table 4.2** This table lists the accuracies obtained on the Balanced-PACS dataset. All these mentioned accuracies are the averages on five different runs. An ERM with an Inception-ResNetV2 backbone was trained for the purpose of this experiment.

| | ACP | APC | CAP | CPA | PAC | PCA | Average |
|---|---|---|---|---|---|---|---|
| Sketch | 82.85 | 83.48 | 83.74 | 84.42 | 79.66 | 79.56 | **83.36** |

| | CPS | CSP | PCS | PSC | SCP | SPC | |
|---|---|---|---|---|---|---|---|
| Art-Painting | 89.05 | 88.42 | 88.62 | 89.16 | 88.87 | 88.33 | **88.70** |

| | APS | ASP | PAS | PSA | SAP | SPA | |
|---|---|---|---|---|---|---|---|
| Cartoon | 81.87 | 81.27 | 76.23 | 77.87 | 82.59 | 82.08 | **79.79** |

| | ACS | ASC | CSA | CAS | SAC | SCA | |
|---|---|---|---|---|---|---|---|
| Photo | 96.23 | 95.87 | 94.01 | 95.21 | 95.21 | 93.42 | **95.37** |

**Table 4.3** This table shows the results obtained across all the permutations in a long tail across training domains. These results were obtained on a single run of these experiments. An ERM with an Inception-ResNetV2 backbone was trained for the purpose of this experiment. The columns in this table resemble the different arrangements of the training domains. For example, considering the test domain Sketch, we have six different permutations possible for the training domains shown as ACP, APC, CAP, CPA, PAC, and PCA in the columns. In these permutations, "P" represents the domain Photo, "A" represents the domain Art-Painting, "C" represents the domain Cartoon, and S represents the domain Sketch.

Further, to show that the order is maintained even across all the permutations of the training domains, with a long-tailed distribution across the training domains, we find the performances across all the permutations of the training domains in a train-test split. For every train-test split of the PACS dataset, we consider one of the domains as the test domain and the remaining three domains as the training domains. Considering the three domain's permutations in the training split, we get six different permutations for every training split. For each of these permutations, we maintain a long-tailed distribution with a common ratio of approximately 0.1 across the permuted training domains. We train an ERM using the same settings as in the previous experiment, i.e., a learning rate of 0.01, Inception-ResNetV2 backbone, and training for 30 epochs. Table 4.3 shows the results of these experiments on a single run. When comparing the average accuracies across the permutations of the training domains, we see that the order of performances remains $Cartoon < Sketch < Art - Painting < Photo$, demonstrating that the model has a high tendency to maintain the relative ordering of the performances, irrespective of the long-tailed

distribution across training domains, along with different permutations.

## 4.3 Additional experiments on PACS

In this section, we will examine some further experiments conducted on PACS. The outcomes of these experiments are expected to enhance our understanding of how ERMs work in the context of DG and provide motivation for further work to the researchers in the field.

### 4.3.1 Experiments showing that the performance of ERMs is agnostic to pre-training

For this experiment, we train a ResNet-18 backbone on both ERM and CDCC based DG settings. For both these settings, we perform the training with five different pre-training criteria, which are a) Without any pre-training, b) pre-trained on Photos, c) pre-trained on Art-Paintings, d) pre-trained on Cartoons and e) pre-trained on Sketch. For all these pre-trained models, we then fine-tune these models for every train-test split in the DG setting using the keep-one-domain-out strategy (one domain is kept out as the target domain, and the remaining ones are kept as training domains). The results of this experiment are reported in 4.4. All the numbers reported in the table are the results of a single run. 30 epochs were run for all the experiments with a learning rate of 0.1.

| Domain | No pre-training | | pre-trained on P | | pre-trained on A | | pre-trained on C | | pre-trained on S | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ERM | CDCC | ERM | CDCC | ERM | CDCC | ERM | CDCC | ERM | CDCC |
| Photo | 60.58 | 63.39 | 62.41 | 64.46 | 58.97 | 59.64 | 60.89 | 61.96 | 62.10 | 63.21 |
| Art-Painting | 48.48 | 49.64 | 55.22 | 55.71 | 50.72 | 53.93 | 46.96 | 47.68 | 50.98 | 51.07 |
| Cartoon | 63.57 | 65.03 | 64.51 | 65.89 | 69.42 | 70.54 | 62.41 | 64.46 | 67.59 | 68.57 |
| Sketch | 72.46 | 75.07 | 72.77 | 74.11 | 74.55 | 75.71 | 68.35 | 68.93 | 75.27 | 77.68 |
| Order | **APCS** | **APCS** | **APCS** | **APCS** | **APCS** | **APCS** | **APCS** | **APCS** | **APCS** | **APCS** |

**Table 4.4** This table reports the accuracies of the PACS dataset with various pre-trainings in both ERM and CDCC settings. Also, the order of performances across the target domain is presented in the last row.

These results show an interesting observation that the order of performances across the target domains remains consistent across all the pre-trainings, showing that the order of performances is agnostic to the pre-trainings. This is an interesting observation that could help researchers analyze and predict

the order of performances across different pre-trainings and, moreover, help get a fair estimate of the expected performances without actually running the experiments.

### 4.3.2 Experiments showing that learning a smoother minima results in a better performance for domain shifts

| | Ar-Cl | Ar-Pr | Ar-Rw | Cl-Ar | Cl-Pr | Cl-Rw | Pr-Ar | Pr-Cl | Pr-Rw | Rw-Ar | Rw-Cl | Rw-Pr | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERM | 44.44 | 54.97 | 64.24 | 47.14 | 54.76 | 57.08 | 50.72 | 44.40 | 65.16 | 58.01 | 50.17 | 71.25 | 55.20 |
| ERM + SAM | 52.97 | 65.17 | 72.50 | 56.28 | 63.84 | 66.01 | 53.69 | 48.61 | 70.71 | 64.85 | 58.28 | 77.27 | 62.52 |

**Table 4.5** The table above compares the performance of ERMs without SAM (using SGD) and with SAM. It shows results for all 12 possible one-to-one domain shifts in the PACS dataset.

In this section, we investigate the impact of achieving smoother minima on the performance of ERMs. Foret et al. [28] introduced a novel gradient update technique called Sharpness Aware Minimization (SAM), which ensures the learning of a smooth minima during optimization. Empirically, it has been shown to lead to enhanced generalization and deliver stable performance, meaning that a model trained with SAM achieves similar results across multiple runs.

To examine whether the advantages of SAM can be leveraged to improve performance and robustness in the face of various domain shifts, we conduct an experiment comparing the performances of standard ERMs to ERMs employing the SAM optimizer (ERM + SAM). We run the experiment for all possible one-to-one domain shifts in the PACS dataset, resulting in a total of 12 distinct domain shifts. The experiment revealed a significant and consistent improvement in performance for ERM + SAM. Refer to 4.5 for the experimental results. These findings strongly indicate that SAM has the potential to be a highly effective optimizer for domain generalization in future DG research trends.

### 4.3.3 Experiments showing that the performance of models pre-trained with self-supervised techniques is inconsistent

In this section, we attempt to understand the effects of self-supervised pre-training techniques on model performance. In the context of Domain Generalization, self-supervised methods like SimCLR [17] and MoCo [18] are expected to allow the model to learn better embeddings. However, when considering semi-supervised pre-trained models, i.e., models that are pre-trained on a dataset using semi-supervised techniques, the performance of these models, after fine-tuning on another dataset (with a linear head), was observed to be inconsistent.

We performed our experiments on the PACS and VLCS datasets using a ResNet-50 backbone. We ran this model with two different pre-training settings: ImageNet pre-trained without MoCo v2 (i.e., using

vanilla ImageNet pre-trained weights without any self-supervised training techniques) and ImageNet pre-trained with MoCo v2. All the experiments were run four times, and the average of these runs is mentioned in 4.6 for PACS and 4.7 for VLCS. The training was conducted with 50 epochs, a batch size of 32, and a learning rate of 0.001 for all these experiments.

| | Cartoon | Art-Painting | Photos | Sketch | Avg |
|---|---|---|---|---|---|
| ImageNet pre-trained w/o MoCo v2 | **83.87** | **84.57** | 94.97 | **83.81** | **86.81** |
| ImageNet pre-trained w/ MoCo v2 | 79.11 | 77.17 | **96.26** | 81.64 | 83.55 |

**Table 4.6** This table presents the results of a ResNet50 model on PACS dataset with ImageNet pre-trained weights w/ and w/o MoCo v2 self-supervision.

| | CALTECH | LABELME | PASCAL | SUN | Avg |
|---|---|---|---|---|---|
| ImageNet pre-trained w/o MoCo v2 | 98.23 | **69.31** | 71.98 | 73.43 | 78.24 |
| ImageNet pre-trained w/ MoCo v2 | **99.22** | 66.75 | **80.86** | **77.51** | **81.09** |

**Table 4.7** This table presents the results of a ResNet50 model on VLCS dataset with ImageNet pre-trained weights w/ and w/o MoCo v2 self-supervision.

It can be observed that the results of the model with and without MoCo pre-training are not consistent. While ImageNet pre-training with MoCo v2 performs better for PACS, it performs worse for VLCS. This results in an interesting observation where weights pre-trained with MoCo perform worse compared to using pre-trained weights without any self-supervised techniques. This behavior of self-supervised pre-training needs to be understood, and we urge the research community to explore this further.

## 4.4 Conclusion

In conclusion, Chapter 4 delves into exploring various aspects that go beyond traditional domain generalization. We investigated class-wise domain generalization and long-tailed domain generalization, showcasing the robustness of CDCC in long-tailed distributions. The experiments also highlighted that the order of performance across test domains remains agnostic to the order of training domains in a long-tailed setting.

Furthermore, we conducted additional experiments on the PACS dataset, demonstrating that the performance of ERMs remains unaffected by pre-training and that learning a smoother minima leads to better performance in the presence of domain shifts. These findings provide valuable insights into the complex nature of domain generalization and offer a foundation for future research in this area. Also, the inconsistent behavior of models pre-trained with self-supervised methods was addressed.

For further exploration, researchers can investigate the extent to which the common ratio can be reduced without compromising performance in a long-tailed setting. Another area worth examining is the non-trivial change in the order of performances when the backbones are altered. By building on the findings of this chapter, we hope to inspire and guide future work in the domain generalization field, leading to more robust and versatile models capable of handling a variety of real-world scenarios.

*Chapter 5*

# Conclusion

In conclusion, this thesis presents a comprehensive analysis of domain generalization (DG), a critical challenge in machine learning that enables models to effectively generalize to new, previously unseen domains. We delved into various aspects of DG, including distribution shift, shortcut learning, representation learning, and data imbalances. Furthermore, we examined key datasets, such as DomainBed, ImageNet Rendition, ImageNet Sketch, and WILDS, and engaged with relevant literature to provide a solid understanding of the current state of the field.

The primary contribution of this thesis is the development of a novel algorithm to quantify the implicit dimensionality of a learned representation and investigate the relationship of this implicit dimensionality with the corresponding DG performance of the learned representation. Through multiple experiments across various backbones, we discovered that models that learn latent representations with lower implicit dimensionality tend to exhibit better DG performance.

To further substantiate this discovery, Cross-Domain Class-Contrastive Learning (CDCC) was proposed as a method for learning sparser representations using supervised contrastive loss. Utilizing CDCC for representation learning resulted in reduced implicit dimensionality and enhanced DG performance, thereby validating our discovery and, moreover, resulting in a new method that exhibits a better DG performance.

Additionally, we ventured beyond traditional DG, exploring emerging challenges such as class-wise DG and long-tailed DG. In our examination of long-tailed DG, we demonstrated that CDCC's performance remains robust to long-tailed distributions. We also showed that the order of performances across test domains is independent of the order of training domains in a long-tailed DG setting (across domains). Furthermore, we conducted experiments using the PACS dataset, revealing that the performance of Empirical Risk Minimizers (ERMs) is not affected by pre-training, and also demonstrated how learning smoother minima using optimizers such as Sharpness Aware Minimization (SAM) can lead to improved performance in a DG setting.

Our goal with these findings is to establish a strong foundation for future research in domain generalization. By illuminating emerging challenges and providing insights into the behavior of various

models and methods, we aim to inspire other researchers to explore novel directions and develop more robust and effective machine learning algorithms for DG.

# Publications

## Thesis Publications

- **Saransh Dave**, Ritam Basu, and Vineet Gandhi; *Cross-Domain Class-Contrastive Learning: Finding Lower Dimensional Representations for Improved Domain Generalization*; **Thirteenth Indian Conference on Computer Vision, Graphics, and Image Processing 2022 (ICVGIP'22)**.

# Bibliography


[1] K. Ahuja, K. Shanmugam, K. Varshney, and A. Dhurandhar. Invariant risk minimization games. *arXiv*, 2020.

[2] K. Akuzawa, Y. Iwasawa, and Y. Matsuo. Adversarial invariant feature learning with accuracy constraint for domain generalization. *arXiv*, 2019.

[3] I. Albuquerque, J. Monteiro, T. H. Falk, and I. Mitliagkas. Adversarial target-invariant representation learning for domain generalization. *arXiv*, 2019.

[4] I. Albuquerque, N. Naik, J. Li, N. Keskar, and R. Socher. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *arXiv*, 2020.

[5] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv:1907.02893*, 2019.

[6] N. Asadi, M. Hosseinzadeh, and M. Eftekhari. Towards shape biased unsupervised representation learning for domain generalization. *arXiv*, 2019.

[7] Y. Balaji, S. Sankaranarayanan, and R. Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NIPS*, 2018.

[8] P. Bandi, O. Geessink, Q. Manson, M. Van Dijk, M. Balkenhol, M. Hermsen, B. E. Bejnordi, B. Lee, K. Paeng, A. Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge. *IEEE Transactions on Medical Imaging*, 2018.

[9] S. Beery, E. Cole, and A. Gjoka. The iwildcam 2020 competition dataset. *arXiv preprint arXiv:2004.10340*, 2020.

[10] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.

[11] G. Blanchard, A. A. Deshmukh, U. Dogan, G. Lee, and C. Scott. Domain generalization by marginal transfer learning. *arXiv*, 2017.

[12] G. Blanchard, G. Lee, and C. Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24:2178–2186, 2011.

[13] D. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, 2019.

[14] F. C. Borlino, A. D'Innocente, and T. Tommasi. Rethinking domain generalization baselines. In *ICPR, 2020*, pages 9227–9233. IEEE, 2021.

[15] V. Bouvier, P. Very, C. Hudelot, and C. Chastagnol. Hidden covariate shift: A minimal assumption for domain adaptation. *arXiv*, 2019.

[16] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019a.

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[18] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[19] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[20] E. David, S. Madec, P. Sadeghi-Tehran, H. Aasen, B. Zheng, S. Liu, N. Kirchgessner, G. Ishikawa, K. Nagasawa, M. A. Badhon, C. Pozniak, B. de Solan, A. Hund, S. C. Chapman, F. Baret, I. Stavness, and W. Guo. Global wheat head detection (gwhd) dataset: a large and diverse dataset of high-resolution rgb-labelled images to develop and benchmark wheat head detection methods. *Plant Phenomics*, 2020, 2020.

[21] E. David, M. Serouart, D. Smith, S. Madec, K. Velumani, S. Liu, X. Wang, F. P. Espinosa, S. Shafiee, I. S. A. Tahir, H. Tsujimoto, S. Nasuda, B. Zheng, N. Kichgessner, H. Aasen, A. Hund, P. Sadhegi-Tehran, K. Nagasawa, G. Ishikawa, S. Dandrifosse, A. Carlier, B. Mercatoris, K. Kuroki, H. Wang, M. Ishii, M. A. Badhon, C. Pozniak, D. S. LeBauer, M. Lilimo, J. Poland, S. Chapman, B. de Solan, F. Baret, I. Stavness, and W. Guo. Global wheat head dataset 2021: an update to improve the benchmarking wheat head localization with more diversity, 2021.

[22] A. A. Deshmukh, Y. Lei, S. Sharma, U. Dogan, J. W. Cutler, and C. Scott. A generalization error bound for multi-class domain generalization. *arXiv*, 2019.

[23] Z. Ding and Y. Fu. Deep domain generalization with structured low-rank constraint. *IEEE Transactions on Image Processing*, 2017.

[24] A. DInnocente and B. Caputo. Domain generalization with domain-specific aggregation modules. In *German Conference on Pattern Recognition*, 2018.

[25] Q. Dou, D. C. de Castro, K. Kamnitsas, and B. Glocker. Domain generalization via model-agnostic learning of semantic features. In *NIPS*, 2019.

[26] C. Fang, Y. Xu, and D. N. Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *ICCV*, 2013.

[27] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[28] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[29] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189. PMLR, 2015.

[30] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016.

[31] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[32] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *IEEE TPAMI*, 2016.

[33] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, pages 2551–2559, 2015.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[35] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Sch"olkopf, and A. Smola. A kernel two-sample test. *JMLR*, 2012.

[36] I. Gulrajani and D. Lopez-Paz. In search of lost domain generalization. *arXiv:2007.01434*, 2020.

[37] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[39] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.

[40] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv:1903.12261*, 2019.

[41] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv:1912.02781*, 2019.

[42] S. Hu, K. Zhang, Z. Chen, and L. Chan. Domain generalization via multidomain discriminant analysis. In *UAI*, 2019.

[43] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[44] W. Hu, G. Niu, I. Sato, and M. Sugiyama. Does distributionally robust supervised learning give robust classifiers? *arXiv*, 2016.

[45] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[46] Z. Huang, H. Wang, E. P. Xing, and D. Huang. Self-challenging improves cross-domain generalization. *ECCV*, 2020.

[47] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling. Diva: Domain invariant variational autoencoders. *arXiv preprint arXiv:1905.10427*, 2019.

[48] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015.

[49] F. D. Johansson, D. Sontag, and R. Ranganath. Support and invertibility in domain-invariant representations. *arXiv*, 2019.

[50] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *ECCV*, 2012.

[51] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *NIPS*, 33:18661–18673, 2020.

[52] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[53] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[55] D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, R. Le Priol, and A. Courville. Out-of-distribution generalization via risk extrapolation (rex). *arXiv*, 2020.

[56] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales. Sequential learning for domain generalization. *arXiv*, 2020.

[57] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, pages 5542–5550, 2017.

[58] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018.

[59] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales. Episodic training for domain generalization. In *ICCV*, 2019a.

[60] H. Li, S. J. Pan, S. Wang, and A. C. Kot. Domain generalization with adversarial feature learning. In *CVPR*, pages 5400–5409, 2018.

[61] Y. Li, M. Gong, X. Tian, T. Liu, and D. Tao. Domain generalization via conditional invariant representations. In *AAAI*, 2018c.

[62] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao. Deep domain generalization via conditional invariant adversarial networks. In *ECCV*, pages 624–639, 2018.

[63] Y. Li, Y. Yang, W. Zhou, and T. M. Hospedales. Feature-critic networks for heterogeneous domain generalization. *arXiv*, 2019b.

[64] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*, 2021.

[65] M. Mancini, S. R. Bulo, B. Caputo, and E. Ricci. Best sources forward: domain generalization through source-specific nets. In *ICIP*, pages 1353–1357. IEEE, 2018.

[66] M. Mancini, S. Rota Bulo, B. Caputo, and E. Ricci. Robust place categorization with deep domain generalization. *IEEE Robotics and Automation Letters*, 2018b.

[67] T. Matsuura and T. Harada. Domain generalization using a mixture of multiple latent domains. *arXiv*, 2019.

[68] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 2012.

[69] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. In *ICCV*, 2017.

[70] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *ICML*, pages 10–18. PMLR, 2013.

[71] K. Muandet, K. Fukumizu, B. Sriperumbudur, B. Scholkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 2017.

[72] H. Nam, H. Lee, J. Park, W. Yoon, and D. Yoo. Reducing domain gap via style-agnostic networks. *arXiv*, 2019.

[73] J. Ni, J. Li, and J. McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[74] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.

[75] J. Peters, P. Buhlmann, and N. Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2016.

[76] V. Piratla, P. Netrapalli, and S. Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *ICML*, 2020.

[77] F. Qiao, L. Zhao, and X. Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020.

[78] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan. Correlation-aware adversarial domain adaptation and generalization. *Pattern Recognition*, 2019a.

[79] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan. Multi-component image translation for deep domain generalization. In *WACV*, 2019b.

[80] V. Raychev, P. Bielik, and M. Vechev. Probabilistic model for code with decision trees. *ACM SIGPLAN Notices*, 2016.

[81] M. Rojas-Carulla, B. Scholkopf, R. Turner, and J. Peters. Invariant models for causal transfer learning. *JMLR*, 2018.

[82] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *ICLR*, 2020.

[83] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing across domains via cross-gradient training. *arXiv*, 2018.

[84] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[85] S. Sivaprasad, A. Goindani, V. Garg, and V. Gandhi. Reappraising domain generalization in neural networks. *arXiv:2110.07981*, 2021.

[86] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

[87] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016.

[88] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[89] J. Taylor, B. Earnshaw, B. Mabey, M. Victors, and J. Yosinski. Rxrx1: An image set for cellular morphological variation across many experimental batches. In *International Conference on Learning Representations (ICLR)*, 2019.

[90] D. Teney, E. Abbasnejad, and A. van den Hengel. Unshuffling data for improved generalization. *arxiv*, 2020.

[91] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, pages 5018–5027, 2017.

[92] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. In *NIPS*, 2018.

[93] H. Wang, S. Ge, Z. Lipton, and E. P. Xing. Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32, 2019.

[94] H. Wang, Z. He, Z. C. Lipton, and E. P. Xing. Learning robust representations by projecting superficial statistics out. *arXiv*, 2019.

[95] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[96] P. Wang, K. Han, X.-S. Wei, L. Zhang, and L. Wang. Contrastive learning based hybrid networks for long-tailed image classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 943–952, 2021.

[97] R. Wang, Z. Wu, Z. Weng, J. Chen, G. Qi, and Y. Jiang. Cross-domain contrastive learning for unsupervised domain adaptation. *CoRR*, abs/2106.05528, 2021.

[98] Y. Wang, H. Li, and A. C. Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP*, 2020.

[99] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang. Adversarial domain adaptation with domain mixup. In *AAAI*, volume 34, pages 6502–6509, 2020.

[100] S. Yan, H. Song, N. Li, L. Zou, and L. Ren. Improve unsupervised domain adaptation with mixup training. *arXiv*, 2020.

[101] C. Yeh, A. Perez, A. Driscoll, G. Azzari, Z. Tang, D. Lobell, S. Ermon, and M. Burke. Using publicly available satellite imagery and deep learning to understand economic well-being in africa. *Nature Communications*, 2020.

[102] Y. Yoshida and T. Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[103] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2100–2110, 2019.

[104] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[105] L. Zhang, X. Wang, D. Yang, T. Sanford, S. Harmon, B. Turkbey, H. Roth, A. Myronenko, D. Xu, and Z. Xu. When unseen domain generalization is unnecessary? rethinking data augmentation. *arXiv*, 2019.

[106] H. Zhao, R. Tachet des Combes, K. Zhang, and G. J. Gordon. On learning invariant representation for domain adaptation. *arXiv*, 2019.

[107] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy. Domain generalization: A survey. *arXiv e-prints*, 2021.

[108] K. Zhou, Y. Yang, T. Hospedales, and T. Xiang. Deep domain-adversarial image generation for domain generalisation. *arXiv preprint arXiv:2003.06054*, 2020.

[109] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain generalization with mixstyle. *ICLR*, 2021.