Toward Robust Machine Learning by Countering Superficial Features

Haohan Wang

CMU-LTI-21-018

Sept. 2021

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Eric P. Xing, Chair
Zachary Lipton
Zico Kolter
Dawn Song (UC Berkeley)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Copyright © 2021 Haohan Wang

Keywords: Robust Machine Learning, Spurious Correlation, Superficial Features

Abstract

Machine learning, especially deep neural networks, has demonstrated remarkable empirical performances over various benchmarks. A potential next step is to extend these empirical successes beyond the i.i.d setting to a more practical scenario where the test data can be collected independently from the training data, while considered as the same task. In other words, how to train a *robust* model with data from one distribution and the test performance will not vary significantly over data from other different but related distributions. While there are many different works devoted to solve this problem of learning robust models from various perspectives, this thesis aims to complement other studies by offering a set of tools for the situation (and under the hypothesis) that one potential issue behind of the model's non-robustness behaviors is the model's tendency to predict through some features, which we refer to as superficial features.

We aim to attack the problem of learning robust models with several technical weapons: we first introduce a line of empirical efforts with numerical successes on different robustness-related benchmarks; we further aim to formally discuss the problem by assuming the challenges lie in the tendency of models' learning of superficial features, which will also lead to a set of principled solutions; we also contribute engineering efforts to deliver a software that allows human to interact with image classification models to improve the model's robustness against superficial features.

In particular, we first hypothesize the underlying challenge of learning robust model lies in the data, and then validate our hypothesis by investigating the model's behavior responding to different copies of the image data.

Building upon our hypothesis, we introduce several new methods for image classification, countering different specific superficial features in the data. The success of these methods are validated as the empirical performances over standard domain generalization image classification tasks.

Further, with the empirical success, we propose to formalize the problem of training a model over data with superficial features. With the knowledge of the superficial features, the formalization leads to a proved bound of the generalization error over the distribution absent of the superficial features. Our formalization can connect to our proposed methods in the previous section well. Our bound will also inspire a new method forgoing the knowledge of superficial features with strong empirical successes.

Finally, to foster the process of building robust models, we introduce a software with GUI that allows users to inspect image classification model's decision process and annotate superficial features exploited by the model.

Contents

Chapter 1

Introduction

1.1 Background & Motivations

In the last decade, machine learning, especially deep neural networks, has accomplished remarkable empirical successes over various benchmark datasets. These achievements, such as paralleling human performances in image classification [e.g., 56, 83] or in natural language understanding [e.g., 115], are so impressive that machine learning, or AI in general, has been widely considered as the next industrial revolution. However, a potential challenge of the industry transition is the observed phenomenon that many machine learning models with noteworthy empirical performances on benchmark datasets often cannot preserve the scores when tested over other datasets that human consider similar, which may be a major bottleneck of deploying the machine learning models to facilitate industrial transitions to be AI-driven. This thesis is built upon this background, aiming of offer a systematic view of training a machine learning model that will be more resilient to the performance drop when tested in other related datasets after deployed.

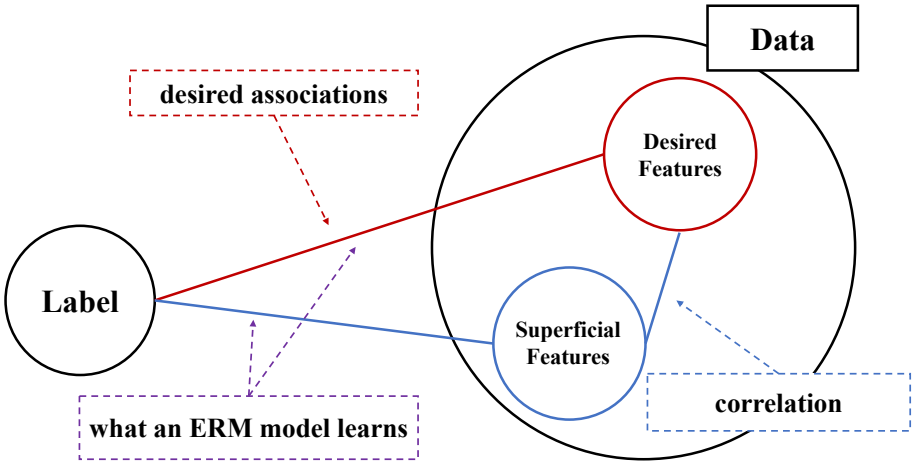


Figure 1.1: The main problem studied in this thesis: how to force the model to learn the desired associations, while the data has superficial correlation and the empirical risk minimizer (ERM) may learn to predict from both of the desired and superficial features.

As the issue of learning robust models may cover a variety of topics that have been explored from various perspectives [e.g., 8, 9, 104, 131, 145], this thesis only studies a slice of the problem with a narrow focus on the situations where the reason of the non-robust behavior as the result of model's learning of "superficial features", a topic also frequently studied under other terminologies such as spurious correlation [155], confounding factors [109], and dataset biases [150]. This main research question is illustrated in Figure 1.1: within a collected dataset, there are desired features that we hope the models to learn, but there are also some other features that are superficially correlated with the label and can be exploited to reduce prediction errors. An empirical risk minimization (ERM) model is not designed to have a preference over one set of features over the other, and may learn superficially correlated features. As a result, an ERM model may see a performance drop when tested with other similar but different datasets: being similar means that other datasets share the same association between the label and the desired features with training data (thus human may consider these datasets similar), and being different means other datasets do not necessarily share the associations between the label and the superficial features with training data.

Many related topics have been widely studied over multiple possible superficial correlation with impressive empirical achievements in various applications, such as computer vision [e.g., 5, 41, 60, 63, 162, 163, 165], natural language understanding [e.g., 55, 105], and computational biology [e.g., 161, 164]. With the proliferation of these related works, we notice that some of these solutions follow a two-step paradigm: first to identify some superficially correlated features that can lead the models to non-robust behaviors, and then to force the models to ignore (i.e., be invariant of) these undesired features. However, we also notice that some of these works tend to reinvent this paradigm to solve each empirical problem.

Therefore, we believe there may be a need of a principled view of these topics. This thesis will first open with empirical observations and methods development, then attempt to offer some a formal discussion of the model's non-robust behaviors as a result of learning the superficial features, which immediately offers a series of principled solutions to the problem.

1.2 Thesis Contributions Overview

We aim to offer an overview of how to learn robust machine learning models by countering superficial features, with the following goals in particular:

- **Central Hypothesis and Empirical Observations:** Chapter 2 defines the scope of this thesis with the central hypothesis of the challenges of learning robust models. We will first reiterate the hypothesis listed in Figure 1.1 and discuss some empirical observations to validate our hypothesis. We will also leverage our empirical observation to explain several interesting machine learning behaviors.
- **Empirical Method Development:** Chapter 3 will focus on the empirical side of this problem. Over the battleground of image classification, the chapter will discuss an armory of methods over deep neural networks to compete with previous SOTA methods. The main evaluation metric is cross-domain test accuracy, with a scenario we propose and refer to as *domain generalization without domain IDs*. This chapter will also contribute a new dataset and evaluation metrics developed to test the robustness of an image classification model.

- **Statistical Support and Principled Solutions:** Chapter 4 aims to build the theoretical support of this problem, proving a new generalization bound when the model is trained over data with superficial correlation. The formalization helps explain the performance drop of when the models are tested with other dataset without the superficial correlation. Also, the formalization hopefully can lead to a set of principled solutions for this problem.
- **Software Development:** Finally, Chapter 5 will contribute a toolbox with graphic user interface that allows users to examine how an image classification model perceives the data and to annotate the superficial features the model exploits. With these annotations, the system can continue to fix the decision process other proposed solutions.

As a disclaimer, we remind the readers that this thesis is built upon the assumption that there are some features in the data that are not beneficial to the model once learned for generalization in the robustness setting. We hope to remind the readers that these features may also be useful in other situations, especially the situations when i.i.d test accuracy is the primary goal, while our evidence to support the hypothesis (see next section) primarily concerns with the robustness settings.

Chapter 2

The Scope of the Problem: the Hypothesis of Superficial Features

In this chapter, we first define the scope of the problems this thesis is devoted to: we aim to offer a set of solutions to learn robust models from multiple different perspectives, but limited to the problems where the main challenges of learning robust models lie in the existence of some other features that are correlated with the label but not considered useful by a human.

This problem has been widely studied under the terminologies such as bias features [150], spurious correlations [155], and confounding factors [109], and one of the most popular example is probably the wolf vs. husky image classification where the snow background features are high correlated with the labels and exploited by the model for classification [127].

In Section 2.1, we first introduce an interesting observation with the perturbation of the frequency domain of the images. With the observation, we argue that even with simple dataset such as CIFAR10, the model may not be able to learn the desired patterns of the images, but learn the superficial features (the high-frequency component of the features) instead. Thus, although the scope of the problem this thesis focuses on is limited, we believe it is fairly important. With the ground built in Section 2.1, we continue to use the technique to further explain several other machine learning phenomenon with the help of superficial features in Section 2.2.

2.1 The Hypothesis and Empirical Validation

Hypothesis: We first reiterate our central hypothesis illustrated in Figure 1.1: within a collected dataset, there are desired features that we hope the models to learn, but there are also some other features that are spuriously correlated with the label and can be exploited to reduce prediction errors. An empirical risk minimization (ERM) model is not designed to have a preference over one set of features over the other, and may learn spuriously correlated features. As a result, an ERM model may see a performance drop when tested with other similar but different datasets: being similar means that other datasets share the same association between the label and the desired features with training data (thus human may consider these datasets similar), and being different means other datasets do not necessarily share the associations between the label and the spurious features with training data.

Validation: To validate this hypothesis, we demonstrate an example, where we perturb the frequency domain of test images and feed the perturbed images back to the model. In conclusion, we demonstrate that there is no explicit tendency that the model will learn the semantic features from the images (*i.e.*, the model sometimes learns the semantic features and sometimes learns the superficial features from the data.) Below are the technical details and the results to support this argument.

Notations: $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes a data sample (the image and the corresponding label). $f(\cdot; \theta)$ denotes a convolutional neural network whose parameters are denoted as θ. We use $\mathcal{H}$ to denote a human model, and as a result, $f(\cdot; \mathcal{H})$ denotes how human will classify the data $\cdot$. $l(\cdot, \cdot)$ denotes a generic loss function (*e.g.*, cross entropy loss). $\alpha(\cdot, \cdot)$ denotes a function evaluating prediction accuracy (for every sample, this function yields 1.0 if the sample is correctly classified, 0.0 otherwise). $d(\cdot, \cdot)$ denotes a function evaluating the distance between two vectors. $\mathcal{F}(\cdot)$ denotes the Fourier transform; thus, $\mathcal{F}^{-1}(\cdot)$ denotes the inverse Fourier transform. We use $\mathbf{z}$ to denote the frequency component of a sample. Therefore, we have $\mathbf{z} = \mathcal{F}(\mathbf{x})$ and $\mathbf{x} = \mathcal{F}^{-1}(\mathbf{z})$.

Notice that Fourier transform or its inverse may introduce complex numbers. In this project, we simply discard the imaginary part of the results of $\mathcal{F}^{-1}(\cdot)$ to make sure the resulting image can be fed into CNN as usual.

Methods We decompose the raw data $\mathbf{x} = \{\mathbf{x}_l, \mathbf{x}_h\}$, where $\mathbf{x}_l$ and $\mathbf{x}_h$ denote the low-frequency component (shortened as LFC) and high-frequency component (shortened as HFC) of $\mathbf{x}$. We have the following four equations:

$$\begin{aligned} \mathbf{z} &= \mathcal{F}(\mathbf{x}), & \mathbf{z}_l, \mathbf{z}_h &= t(\mathbf{z}; r), \\ \mathbf{x}_l &= \mathcal{F}^{-1}(\mathbf{z}_l), & \mathbf{x}_h &= \mathcal{F}^{-1}(\mathbf{z}_h), \end{aligned}$$

where $t(\cdot; r)$ denotes a thresholding function that separates the low and high frequency components from $\mathbf{z}$ according to a hyperparameter, radius r.

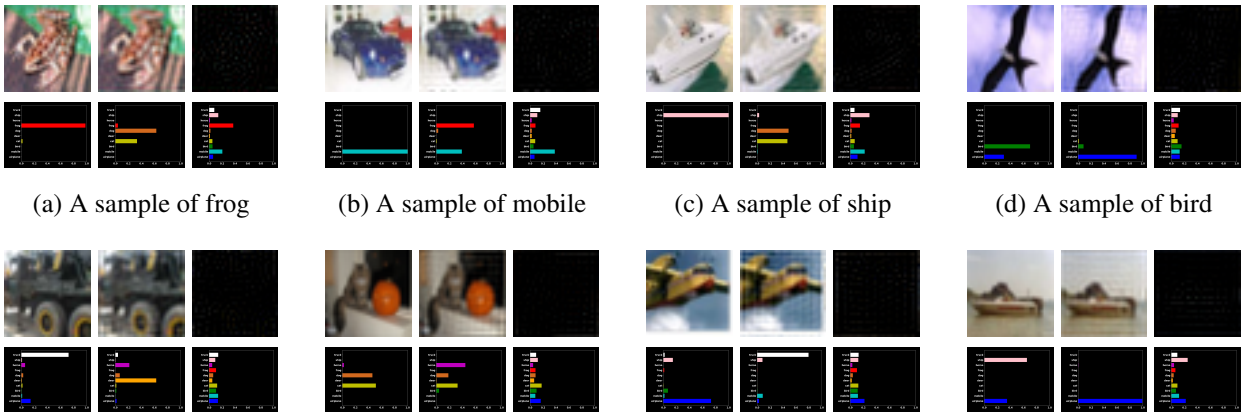
To define $t(\cdot; r)$ formally, we first consider a grayscale (one channel) image of size $n \times n$ with $\mathcal{N}$ possible pixel values (in other words, $\mathbf{x} \in \mathcal{N}^{n \times n}$), then we have $\mathbf{z} \in \mathcal{C}^{n \times n}$, where $\mathcal{C}$ denotes the complex number. We use $\mathbf{z}(i, j)$ to index the value of $\mathbf{z}$ at position (i, j), and we use c_i, c_j to denote the centroid. We have the equation $\mathbf{z}_l, \mathbf{z}_h = t(\mathbf{z}; r)$ formally defined as:

$$\mathbf{z}_l(i,j) = \begin{cases} \mathbf{z}(i,j), & \text{if } d((i,j),(c_i,c_j)) \leqslant r \\ 0, & \text{otherwise} \end{cases},$$

$$\mathbf{z}_h(i,j) = \begin{cases} 0, & \text{if } d((i,j),(c_i,c_j)) \leqslant r \\ \mathbf{z}(i,j), & \text{otherwise} \end{cases}$$

We consider $d(\cdot, \cdot)$ in $t(\cdot; r)$ as the Euclidean distance in this section. If $\mathbf{x}$ has more than one channel, then the procedure operates on every channel of pixels independently.

Results: With the above setup, we can show that sometimes, the model captures the high-frequency component instead of the semantic features in image classification. For example,



(a) A sample of frog (b) A sample of mobile (c) A sample of ship (d) A sample of bird

(e) A sample of truck (f) A sample of cat (g) A sample of airplane (h) A sample of ship

Figure 2.1: Eight testing samples selected from CIFAR10 that help explain that CNN can capture the high-frequency image: the model (ResNet18) correctly predicts the original image (1st column in each panel) and the high-frequency reconstructed image (3rd column in each panel), but incorrectly predict the low-frequency reconstructed image (2nd column in each panel). The prediction confidences are also shown. The frequency components are split with $r = 12$. Details of the experiment will be introduced later.

Figure 2.1 shows the prediction results of eight testing samples from CIFAR10 data set, together with the prediction results of the high and low-frequency component counterparts. For these examples, the prediction outcomes are almost entirely determined by the high-frequency components of the image, which are barely perceivable to human. On the other hand, the low-frequency components, which almost look identical to the original image to human, are predicted to something distinctly different by the model.

Discussion:

Remark 1. With an assumption (referred to as A1) that presumes "only $\mathbf{x}_l$ is perceivable to human, but both $\mathbf{x}_l$ and $\mathbf{x}_h$ are perceivable to a CNN," we have:

$$\mathbf{y} := f(\mathbf{x}; \mathcal{H}) = f(\mathbf{x}_l; \mathcal{H}),$$

but when a CNN is trained with

$$\arg\min_{\theta} l(f(\mathbf{x}; \theta), \mathbf{y}),$$

which is equivalent to

$$\arg\min_{\theta} l(f(\{\mathbf{x}_l, \mathbf{x}_h\}; \theta), \mathbf{y}),$$

CNN may learn to exploit $\mathbf{x}_h$ to minimize the loss. As a result, CNN's generalization behavior appears unintuitive to a human. □

Notice that "CNN may learn to exploit $\mathbf{x}_h$" differs from "CNN overfit" because $\mathbf{x}_h$ can contain more information than sample-specific idiosyncrasy, and these more information can be generalizable across training, validation, and testing sets, but are just imperceptible to a human.

As Assumption A1 has been demonstrated to hold in some cases (*e.g.*, in Figure 2.1), we believe Remark 1 can serve as one of the explanations to CNN's generalization behavior. For example, the adversarial examples [49, 145] can be generated by perturbing $\mathbf{x}_h$; the capacity of CNN in reducing training error to zero over label shuffled data [178] can be seen as a result of exploiting $\mathbf{x}_h$ and overfitting sample-specific idiosyncrasy. Further, we attempt to leverage our techniques to offer explanations to several other machine learning methods.

2.2 Other Empirical Observations

The above technique also conveniently allows us to analyzer several other machine learning generalization properties, and it seems many generalization mysteries can be credited to the fact that there are multiple predictive signals in the data.

2.2.1 Rethinking Data before Rethinking Generalization

Hypothesis Our first aim is to offer some intuitive explanations to the empirical results observed in [178]: neural networks can easily fit label-shuffled data. While we have no doubts that neural networks are capable of memorizing the data due to its capacity, the interesting question arises: "if a neural network can easily memorize the data, why it cares to learn the generalizable patterns out of the data, in contrast to directly memorizing everything to reduce the training loss?"

Within the perspective introduced in Remark 1, our hypothesis is as follows: Despite the same outcome as a minimization of the training loss, the model considers different level of features in the two situations:

- In the original label case, the model will first pick up LFC, then gradually pick up the HFC to achieve higher training accuracy.
- In the shuffled label case, as the association between LFC and the label is erased due to shuffling, the model has to memorize the images when the LFC and HFC are treated equally.

Experiments We set up the experiment to test our hypothesis. We use ResNet-18 [57] for CIFAR10 dataset [82] as the base experiment. The vanilla set-up, which we will use for the rest of this section, is to run the experiment with 100 epoches with the ADAM optimizer [79] with learning rate set to be 10^{-4} and batch size set to be 100, when weights are initialized with Xavier initialization [46]. Pixels are all normalized to be $[0, 1]$. All these experiments are repeated in MNIST [27], FashionMNIST [172], and a subset of ImageNet [26]. These efforts are reported in the Appendix. We train two models, with the natural label setup and the shuffled label setup, denote as $\mathsf{M}_{\mathsf{natural}}$ and $\mathsf{M}_{\mathsf{shuffle}}$, respectively; the $\mathsf{M}_{\mathsf{shuffle}}$ needs 300 epoches to reach a comparative training accuracy. To test which part of the information the model picks up, for any $\mathbf{x}$ in the

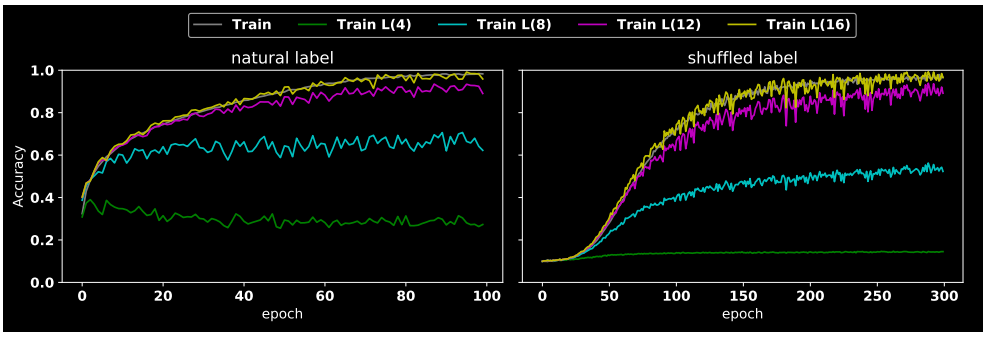


Figure 2.2: Training curves of the original label case (100 epoches) and shuffled label case (300 epoches), together plotted with the low-frequent counterpart of the images. All curves in this figure are from train samples.

Table 2.1: We test the generalization power of LFC and HFC by training the model with $\mathbf{x}_l$ or $\mathbf{x}_h$ and test on the original test set.

LFC			HFC		
r	train acc.	test acc.	r	train acc.	test acc.
4	0.9668	0.6167	4	0.9885	0.2002
8	0.9786	0.7154	8	0.9768	0.092
12	0.9786	0.7516	12	0.9797	0.0997
16	0.9839	0.7714	16	0.9384	0.1281

training set, we generate the low-frequency counterparts $\mathbf{x}_l$ with r set to $4, 8, 12, 16$ respectively. We test the how the training accuracy changes for these low-frequency data collections along the training process.

The results are plotted in Figure 2.2. The first message is the $\mathsf{M}_{\mathsf{shuffle}}$ takes a longer training time than $\mathsf{M}_{\mathsf{natural}}$ to reach the same training accuracy (300 epoches vs. 100 epoches), which suggests that memorizing the samples as an "unnatural" behavior in contrast to learning the generalizable patterns. By comparing the curves of the low-frequent training samples, we notice that $\mathsf{M}_{\mathsf{natural}}$ learns more of the low-frequent patterns (*i.e.*, when r is 4 or 8) than $\mathsf{M}_{\mathsf{shuffle}}$. Also, $\mathsf{M}_{\mathsf{shuffle}}$ barely learns any LFC when $r = 4$, while on the other hand, even at the first epoch, $\mathsf{M}_{\mathsf{natural}}$ already learns around 40% of the correct LFC when $r = 4$. This disparity suggests that when $\mathsf{M}_{\mathsf{natural}}$ prefers to pick up the LFC, $\mathsf{M}_{\mathsf{shuffle}}$ does not have a preference between LFC vs. HFC.

If a model can exploit multiple different sets of signals, then why $\mathsf{M}_{\mathsf{natural}}$ prefers to learn LFC that happens to align well with the human perceptual preference? While there are explanations suggesting neural networks' tendency towards simpler functions [124], we conjecture that this is simply because, since the data sets are organized and annotated by human, the LFC-label association is more "generalizable" than the one of HFC: picking up LFC-label association will lead to the steepest descent of the loss surface, especially at the early stage of the training.

To test this conjecture, we repeat the experiment of $\mathsf{M}_{\mathsf{natural}}$, but instead of the original train set, we use the $\mathbf{x}_l$ or $\mathbf{x}_h$ (normalized to have the standard pixel scale) and test how well the model can perform on original test set. Table 2.1 suggests that LFC is much more "generalizable" than

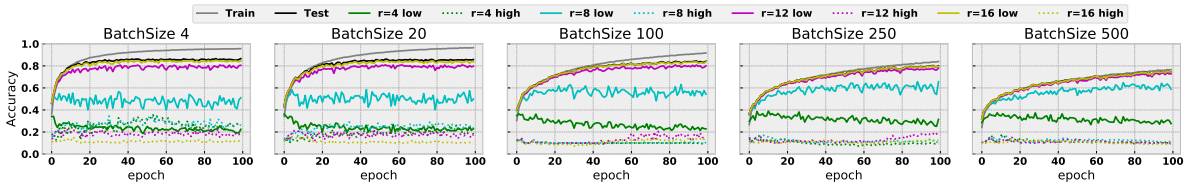


Figure 2.3: Plots of accuracy of different epoch sizes along the epoches for train, test data, as well as LFC and HFC with different radii.

HFC. Thus, it is not surprising if a model first picks up LFC as it leads to the steepest descent of the loss surface.

A Remaining Question Finally, we want to raise a question: The coincidental alignment between networks' preference in LFC and human perceptual preference might be a simple result of the "survival bias" of the many technologies invented one of the other along the process of climbing the ladder of the state-of-the-art. In other words, the almost-100-year development process of neural networks functions like a "natural selection" of technologies [159]. The survived ideas may happen to match the human preferences, otherwise, the ideas may not even be published due to the incompetence in climbing the ladder.

However, an interesting question will be how well these ladder climbing techniques align with the human visual preference. We offer to evaluate these techniques with our frequency tools.

2.2.2 Training Techniques

We continue to reevaluate the techniques that helped in climbing the ladder of state-of-the-art accuracy. We evaluate these techniques to test the generalization performances towards LFC and HFC. Many renowned techniques in the ladder of accuracy seem to exploit HFC more or less.

Comparison of Different techniques We test multiple techniques by inspecting the prediction accuracy over LFC and HFC with multiple choices of r along the training process and plot the training curves.

Batch Size: We then investigate how the choices of batch size affect the generalization behaviors. We plot the results in Figure 2.3. As the figure shows, smaller batch size appears to excel in improving training and testing accuracy, while bigger batch size seems to stand out in closing the generalization gap. Also, it seems the generalization gap is closely related to the model's tendency in capturing HFC: models trained with bigger epoch sizes are more invariant to HFC and introduce smaller differences in training accuracy and testing accuracy. The observed relation is intuitive because the smallest generalization gap will be achieved once the model behaves like a human (because it is the human who annotate the data).

The observation in Figure 2.3 also chips in the discussion in the previous section about "generalizable" features. Intuitively, with bigger epoch size, the features that can lead to steepest

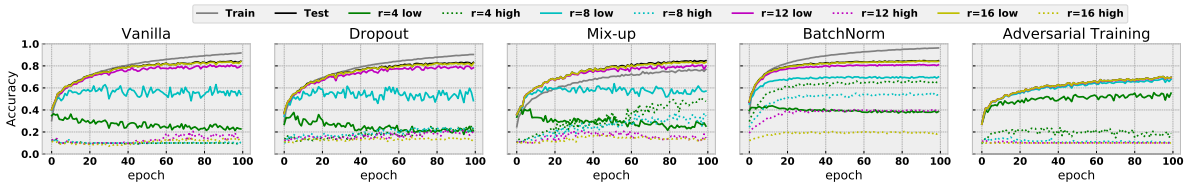


Figure 2.4: Plots of accuracy of different techniques along the epoches for train, test data, as well as LFC and HFC with different radii.

descent of the loss surface are more likely to be the "generalizable" patterns of the data, which are LFC.

techniques: We also test how different training methods react to LFC and HFC, including

- Dropout [65]: A heuristic that drops weights randomly during training. We apply dropout on fully-connected layers with $p = 0.5$.
- Mix-up [179]: A heuristic that linearly integrate samples and their labels during training. We apply it with standard hyperparameter $\alpha = 0.5$.
- BatchNorm [71]: A method that perform the normalization for each training mini-batch to accelerate Deep Network training process. It allows us to use a much higher learning rate and reduce overfitting, similar with Dropout. We apply it with setting scale γ to 1 and offset β to 0.
- Adversarial Training [104]: A method that augments the data through adversarial examples generated by a threat model during training. It is widely considered as one of the most successful adversarial robustness (defense) method. Following the popular choice, we use PGD with $\epsilon = 8/255$ ($\epsilon = 0.03$) as the threat model.

We illustrate the results in Figure 2.4, where the first panel is the vanilla set-up, and then each one of the four techniques are tested in the following four panels.

Dropout roughly behaves similarly to the vanilla set-up in our experiments. Mix-up delivers a similar prediction accuracy, however, it catches much more HFC, which is probably not surprising because the mix-up augmentation does not encourage anything about LFC explicitly, and the performance gain is likely due to attention towards HFC.

Adversarial training mostly behaves as expected: it reports a lower prediction accuracy, which is likely due to the trade-off between robustness and accuracy. It also reports a smaller generalization gap, which is likely as a result of picking up "generalizable" patterns, as verified by its invariance towards HFC (*e.g.*, $r = 12$ or $r = 16$). However, adversarial training seems to be sensitive to the HFC when $r = 4$, which is ignored even by the vanilla set-up.

The performance of BatchNorm is notable: compared to the vanilla set-up, BatchNorm picks more information in both LFC and HFC, especially when $r = 4$ and $r = 8$. This BatchNorm's tendency in capturing HFC is also related to observations that BatchNorm encourages adversarial vulnerability [37].

Other Tests: We have also tested other techniques or methods by only changing along one dimension while the rest is fixed the same as the vanilla set-up.

Model architecture: We tested LeNet [88], AlexNet [83], VGG [140], and ResNet [57]. The ResNet architecture seems advantageous toward previous inventions at different levels: it reports

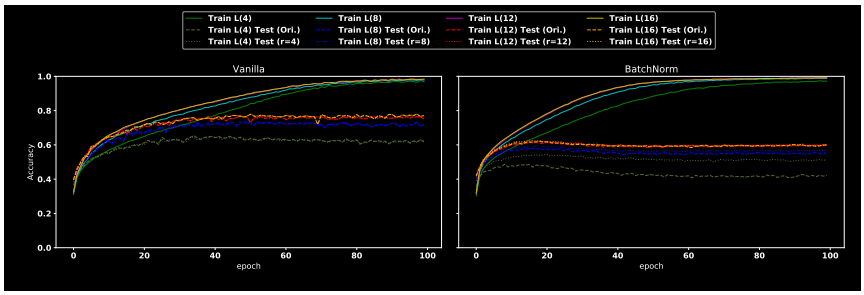


Figure 2.5: Comparison of models with vs. without BatchNorm trained with LFC data.

better vanilla test accuracy, smaller generalization gap (difference between training and testing accuracy), and a weaker tendency in capturing HFC.

Optimizer: We tested SGD, ADAM [79], AdaGrad [33], AdaDelta [177], and RMSprop. We notice that SGD seems to be the only one suffering from the tendency towards significantly capturing HFC, while the rest are on par within our experiments.

A hypothesis on Batch Normalization Based on the observation, we hypothesized that one of BatchNorm's advantage is, through normalization, to align the distributional disparities of different predictive signals. For example, HFC usually shows smaller magnitude than LFC, so a model trained without BatchNorm may not easily pick up these HFC. Therefore, the higher convergence speed may also be considered as a direct result of capturing different predictive signals simultaneously.

To verify this hypothesis, we compare the performance of models trained with vs. without BatchNorm over LFC data and plot the results in Figure 2.5.

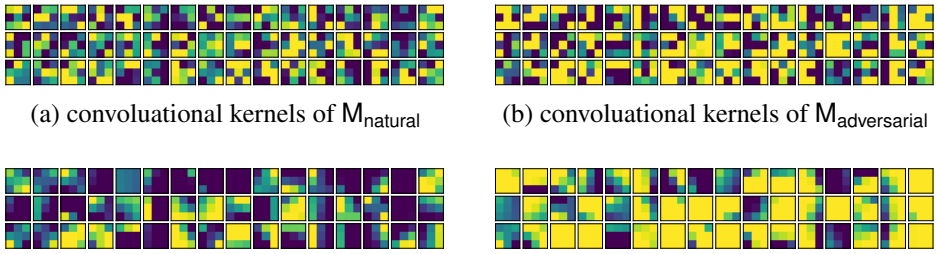
As Figure 2.5 shows, when the model is trained with only LFC, BatchNorm does not always help improve the predictive performance, either tested by original data or by corresponding LFC data. Also, the smaller the radius is, the less the BatchNorm helps. Also, in our setting, BatchNorm does not generalize as well as the vanilla setting, which may raise a question about the benefit of BatchNorm.

However, BatchNorm still seems to at least boost the convergence of training accuracy. Interestingly, the acceleration is the smallest when $r = 4$. This observation further aligns with our hypothesis: if one of BatchNorm's advantage is to encourage the model to capture different predictive signals, the performance gain of BatchNorm is the most limited when the model is trained with LFC when $r = 4$.

2.2.3 Adversarial Attack & Defense

As one may notice, our observation of HFC can be directly linked to the phenomenon of "adversarial example": if the prediction relies on HFC, then perturbation of HFC will significantly alter the model's response, but such perturbation may not be observed to human at all, creating the unintuitive behavior of neural networks.

This section is devoted to study the relationship between adversarial robustness and model's tendency in exploiting HFC.



(a) convoluational kernels of $\mathsf{M}_{\text{natural}}$ (b) convoluational kernels of $\mathsf{M}_{\text{adversarial}}$

(c) convoluational kernels of $\mathsf{M}_{\text{natural}}(r{=}1.0)$ (d) convoluational kernels of $\mathsf{M}_{\text{adversarial}}(r{=}1.0)$

Figure 2.6: Visualization of convolutional kernels (16 kernels each channel $\times$ 3 channels at the first layer) of models.

Kernel Smoothness vs. Image Frequency As convolutional theorem [14] states, the convolution operation of images is equivalent to the element-wise multiplication of image frequency domain. Therefore, roughly, if a convolutional kernel has negligible weight at the high-end of the frequency domain, it will weigh HFC accordingly. This may only apply to the convolutional kernel at the first layer because the kernels at higher layer do not directly with the data, thus the relationship is not clear.

Therefore, we argue that, to push the model to ignore the HFC, one can consider to force the model to learn the convolutional kernels that have only negligible weights at the high-end of the frequency domain.

Intuitively (from signal processing knowledge), if the convolutional kernel is "smooth", which means that there is no dramatics fluctuations between adjacent weights, the corresponding frequency domain will see a negligible amount of high-frequency signals. The connections have been mathematically proved [121, 148], but these proved exact relationships are out of the scope of this section.

Robust Models Have Smooth Kernels To understand the connection between "smoothness" and adversarial robustness, we visualize the convolutional kernels at the first layer of the models trained in the vanilla manner ($\mathsf{M}_{\text{natural}}$) and trained with adversarial training ($\mathsf{M}_{\text{adversarial}}$) in Figure 2.6 (a) and (b).

Comparing Figure 2.6(a) and Figure 2.6(b), we can see that the kernels of $\mathsf{M}_{\text{adversarial}}$ tend to show a more smooth pattern, which can be observed by noticing that the adjacent weights of kernels of $\mathsf{M}_{\text{adversarial}}$ tend to share the same color. The visualization may not be very clear because the convolutional kernel is only [3 $\times$ 3] in ResNet, the message is delivered more clearly in Appendix with other architecture when the first layer has kernel of the size [5 $\times$ 5].

Smoothing Kernels Improves Adversarial Robustness The intuitive argument and empirical findings directly lead to a question of whether we can improve the adversarial robustness of models by smoothing the convolutional kernels at the first layer.

Following the discussion, we introduce an extremely simple method that appears to improve the adversarial robustness against FGSM [49] and PGD [85]. For a convolutional kernel $\mathbf{w}$, we

	Clean	FGSM			PGD		
		$\epsilon = 0.03$	$\epsilon = 0.06$	$\epsilon = 0.09$	$\epsilon = 0.03$	$\epsilon = 0.06$	$\epsilon = 0.09$
$\mathsf{M}_{\mathsf{natural}}$	**0.856**	0.107	0.069	0.044	0.003	0.002	0.002
$\mathsf{M}_{\mathsf{natural}}(\rho = 0.10)$	0.815	0.149	0.105	0.073	0.009	0.002	0.001
$\mathsf{M}_{\mathsf{natural}}(\rho = 0.25)$	0.743	0.16	0.11	0.079	0.021	0.005	0.005
$\mathsf{M}_{\mathsf{natural}}(\rho = 0.50)$	0.674	0.17	0.11	0.083	0.031	0.016	0.014
$\mathsf{M}_{\mathsf{natural}}(\rho = 1.0)$	0.631	**0.171**	**0.14**	**0.127**	**0.086**	**0.078**	**0.078**
$\mathsf{M}_{\mathsf{adversarial}}$	**0.707**	**0.435**	**0.232**	0.137	**0.403**	0.138	0.038
$\mathsf{M}_{\mathsf{adversarial}}(\rho = 0.10)$	0.691	0.412	0.192	0.109	0.379	0.13	0.047
$\mathsf{M}_{\mathsf{adversarial}}(\rho = 0.25)$	0.667	0.385	0.176	0.097	0.352	0.116	0.04
$\mathsf{M}_{\mathsf{adversarial}}(\rho = 0.50)$	0.653	0.365	0.18	0.106	0.334	0.121	0.062
$\mathsf{M}_{\mathsf{adversarial}}(\rho = 1.0)$	0.638	0.356	0.223	**0.186**	0.337	**0.175**	**0.131**

Table 2.2: Prediction performance of models against different adversarial attacks with different ϵ.

use i and j to denote its column and row indices, thus $\mathbf{w}_{i,j}$ denotes the value at i^{th} row and j^{th} column. If we use $\mathcal{N}(i,j)$ to denote the set of the spatial neighbors of (i,j), our method is simply:

$$\mathbf{w}_{i,j} = \mathbf{w}_{i,j} + \sum_{(h,k)\in\mathcal{N}(i,j)} \rho \mathbf{w}_{h,k}, \tag{2.1}$$

where ρ is a hyperparameter of our method. We fix $\mathcal{N}(i,j)$ to have eight neighbors. If (i,j) is at the edge, then we simply generate the out-of-boundary values by duplicating the values on the boundary.

In other words, we try to smooth the kernel through simply reducing the adjacent differences by mixing the adjacent values. The method barely has any computational load, but appears to improve the adversarial robustness of $\mathsf{M}_{\mathsf{natural}}$ and $\mathsf{M}_{\mathsf{adversarial}}$ towards FGSM and PGD, even when $\mathsf{M}_{\mathsf{adversarial}}$ is trained with PGD as the threat model.

In Figure 2.6, we visualize the convolutional kernels with our method applied to $\mathsf{M}_{\mathsf{natural}}$ and $\mathsf{M}_{\mathsf{adversarial}}$ with $\rho = 1.0$, denoted as $\mathsf{M}_{\mathsf{natural}}(\rho = 1.0)$ and $\mathsf{M}_{\mathsf{adversarial}}(\rho = 1.0)$, respectively. As the visualization shows, the resulting kernels tend to show a significantly smoother pattern.

We test the robustness of the models smoothed by our method against FGSM and PGD with different choices of ϵ, where the maximum of perturbation is 1.0. As Table 2.2 shows, when our smoothing method is applied, the performance of clean accuracy directly plunges, but the performance of adversarial robustness improves. In particular, our method helps when the perturbation is allowed to be relatively large. For example, when $\epsilon = 0.09$ (roughly $23/255$), $\mathsf{M}_{\mathsf{natural}}(\rho = 1.0)$ even outperforms $\mathsf{M}_{\mathsf{adversarial}}$. In general, our method can easily improve the adversarial robustness of $\mathsf{M}_{\mathsf{natural}}$, but can only improve upon $\mathsf{M}_{\mathsf{adversarial}}$ in the case where ϵ is larger, which is probably because the $\mathsf{M}_{\mathsf{adversarial}}$ is trained with PGD($\epsilon = 0.03$) as the threat model.

2.2.4 Similar Properties in Object Detection

We aim to explore more than image classification tasks. We investigate the disparity between CNN and human in the object detection task. We use RetinaNet [100] with ResNet50 [57] +

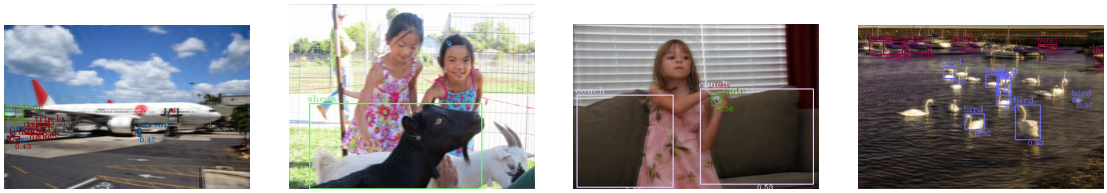


Figure 2.7: Some objects are recognized worse (lower MAP scores) when the experiments are repeated with low-frequent images. Marked objects are the ones that induce differences (objects that are recognized with the same MAP in original images and low-frequent images are not marked).

Figure 2.8: Some objects are recognized better (higher MAP scores) when the experiments are repeated with low-frequent images. Marked objects are the ones that induce differences (objects that are recognized with the same MAP in original images and low-frequent images are not marked).

FPN [99] as the backbone. We train the model with COCO detection train set [98] and perform inference in its validation set, which includes 5000 images, and achieve an MAP of 35.6%.

Then we choose $r = 128$ and maps the images into $\mathbf{x}_l$ and $\mathbf{x}_h$ and test with the same model and get 27.5% MAP with LFC and 10.7% MAP with HFC. The performance drop from 35.6% to 27.5% intrigues us so we further study whether the same drop should be expected from human.

Performance Drop on LFC The performance drop from the $\mathbf{x}$ to $\mathbf{x}_l$ may be expected because $\mathbf{x}_l$ may not have the rich information from the original images when HFC are dropped. In particular, different from image classification, HFC may play a significant role in depicting some objects, especially the smaller ones.

For example, Figure 2.7 illustrates a few examples, where some objects are recognized worse in terms of MAP scores when the input images are replaced by its low-frequent counterpart. This disparity may be expected because the low-frequent images tend to be blurry and some objects may not be clear to a human either (as the left top image represents). However, we also notice multiple images with object recognized inferior to the case of the original images when the low-frequent images maintain a reasonable amount of information.

Performance Gain on LFC However, the disparity gets interesting when we inspect the performance gap in the opposite direction. We identified 1684 images that for each of these images, the some objects are recognized better (high MAP scores) in comparison to the original images.

The results are shown in Figure 2.8. There seems no apparent reasons why these objects

are recognized better in low-frequent images, when inspected by human. These observations strengthen our argument in the perceptual disparity between CNN and human also exist in more advanced computer vision tasks other than image classification.

2.2.5 Discussion

Are high-frequency components just noises? To answer this question, we experiment with another frequently used image denoising method: truncated singular value decomposition (SVD) (*e.g.*, [103]). We first decompose the image with SVD, then instead of separating the image into LFC and HFC, we separate the image into one reconstructed with dominant singular values (the ones with bigger absolute values) and one reconstructed with trailing singular values (the ones with smaller absolute values). With this set-up, we find much fewer images supporting the story in Figure 2.1. Our observations suggest the signal CNN exploit is more than just random "noises".

Other related CNN-analysis works adopting the Fourier transform technique: This work is inspired by empirical observations showing that a CNN has a tendency in learning superficial statistics [75, 163]. Previous work showed the large generalization gap between images of different frequency-domain perturbations of the same model [75]. Guo *et al.* showed that the adversarial attack is particular effective if the perturbations are constrained in low-frequent space [52], which was further analyzed to show the low-frequent perturbations are perceivable to human [137].

2.3 Remarks on the Hypothesis and Its Potential Implications

We investigated how image frequency spectrum affects the generalization behavior of CNN, leading to multiple interesting explanations of the generalization behaviors of neural networks from a new perspective: there are multiple signals in the data, and not all of them align with human's visual preference. Even in simple dataset such as CIFAR10, we can observe the problems of superficial features and its impact on learning the models that can align well with the human's visual preference. Together with many other works that have credited the lack of robustness to the data features [63, 70], we hope this chapter can serve as a foundation of the rest the chapters in discussing the existence of superficial features and the importance in countering them.

Chapter 3

Methods to Learn Robust Models by Countering Superficial Features

This chapter aims to introduce a set of methods to learn robust models by countering superficial features. These methods are validated by empirical performances in the competition of SOTA leaderboards, mainly in cross-domain image classification setup.

In particular, we propose to address the problem with the two following perspectives:

- Following the celebrated power of data augmentation, we continue to ask what are the consistency loss we can use to train a robust model with augmented data. We will also formalize a new criterion, namely "invariance", and show that our method encourages the learning of the invariance (Section 3.2).
- On the other hand, we will demonstrate two techniques that are specifically designed to predict through image texture and image patch respectively, so that the model can further leverage these techniques to discard these spurious information (Section 3.3.3 and 3.3.2).

In addition to the methods above, the chapter will also introduce a new dataset, the ImageNet-Sketch, for cross-domain image classification evaluation (Section 3.4).

The main test bed of these methods is cross-domain image classification, with a setting we invent and refer to as *domain generalization without domain IDs*. Then the following sections will focus on each concrete problem, each with a full-fledged structure of possible additional background and notations, method introduction, empirical results, and discussions when necessary. ImageNet-Sketch will be used in these experiments but the details of the dataset will be introduced in the last section.

3.1 Problem Setup

Nowadays, deep neural networks have exhibited remarkable empirical results over various computer vision tasks, yet these impressive performances seem unmet when the models are tested with the samples in irregular qualities [166] (*i.e.*, out-of-domain data, samples collected from the distributions that are similar to, but different from the distributions of the training samples). To account for this discrepancy, technologies have been invented under the domain adaptation regime [8, 9], where the goal is to train a model with data from the source domain (*i.e.*, the distri-

bution of the training samples) and get reasonably good predictive performance with data from the target domain (*i.e.*, the distribution of the testing samples) [22, 167]. Some data from target domain are usually available during training, either labelled or unlabelled.

Further, as the influence of machine learning increases, the industry starts to demand the models that can be applied to the domains that are not seen during the training phase. Domain generalization [114], as an extension of domain adaptation, has been studied as a response. The main goal here is to train a model from a collection of distributions, and test the predictive performance in an unseen distribution. This setup does not require data from test domain to be available during training, but it usually requires the domain identifiers of the training data.

However, we notice that the setup can be improved to be closer to the industry scenario. Therefore, we extend the problem to ask how to train a model that generalizes to an arbitrary domain with only the training samples, but not the corresponding domain information, as these domain information may not be available in the real world [163]. Most of the remaining sections build upon this set-up and aims to train a model from multiple distributions without domain information and to empirically perform well on unseen domains.

3.2 Data Augmentation & Consistency Loss

3.2.1 Background

Recent advances in deep learning has delivered remarkable empirical performance over *i.i.d* test data, and the community continues to investigate the more challenging and realistic scenario when models are tested in robustness over non-*i.i.d* data [*e.g.*, 9, 145]. Recent studies suggest that one challenge is the model's tendency in capturing undesired signals [41, 77, 166], thus combating this tendency may be a key to learning robust models.

To help models discard the undesired signals, augmentation (*i.e.*, diluting the undesired signals of training samples by applying transformations to existing examples) is often used [*e.g.*, 47, 62]. It is probably one of the most common methods to improve model's performance. Given its wide usage, in this section, we seek to answer the question: *how should we train with augmented samples so that the assistance of augmentation can be taken to the fullest extent to learn robust and invariant models?*

To answer this, we first conduct a range of experiments over image classification benchmarks to evaluate how popular variants of consistency loss contribute to learning robust and invariant models. We test for accuracy, robustness, and invariance, for which we propose a new test procedure. Our empirical study favors squared ℓ_2 norm. Further, as our experiments may not be able to exhaust all options, we complement the empirical study with an analysis of the generalization error bounds of models trained with augmented data and consistency loss. Our theoretical study indicates the optimal choice to be ℓ_1 norm (under assumptions introduced later). With the belief that the disparity can be explained by the difficulties in passing the gradient of ℓ_1 norm in practice, we believe our formal derivations can support the empirical results well. Our contributions are:

- With a new invariance test, we argue that while a vanilla training with augmented data can improve robustness, consistency loss is necessary to regularize the model to learn

representations invariant to the augmentation function, and squared ℓ_2 norm seems to be the best choice as assessed by a variety of empirical evaluations (§3.2.4).

- On the theoretical end, we formalize a generalization error bound for models trained with consistency loss and augmented data, studying the worst-case expected risk over unseen data when samples are allowed to be transformed according to a function in a family (§3.2.5).
- We test the method we identified (squared ℓ_2 norm consistency loss) in multiple robust machine learning scenarios. We believe the fact a generic approach can compete with methods specially designed for different scenarios can endorse its empirical strength (§3.2.6).

3.2.2 Related Work & Key Differences

Data augmentation has been used effectively for years. Tracing back to the earliest convolutional neural networks, we notice that even the LeNet applied on MNIST dataset has been boosted by mixing the distorted images to the original ones [88]. Later, the rapidly growing machine learning community has seen a proliferate development of data augmentation techniques (*e.g.*, flipping, rotation, blurring *etc.*) that have helped models climb the ladder of the state-of-the-art (one may refer to relevant survey [139] for details). Recent advances expanded the conventional concept of data augmentation and invented multiple new approaches to learn accurate and robust models [66, 68, 69, 73, 78, 122, 142, 147, 153, 168, 173, 176, 184, 188, 190]. Among these, the most relevant one to this section will be to generate the samples (with constraint) that maximize the training loss along training [36], which is widely accepted as adversarial training [104].

While the above works mainly discuss how to generate the augmented samples, in this section, we mainly answer the question about how to train the models with augmented samples. For example, instead of directly mixing augmented samples with the original samples, one can consider regularizing the representations (or outputs) of original samples and augmented samples to be close under a distance metric (also known as a consistency loss). Many concrete ideas have been explored in different contexts. For example, ℓ_2 distance and cosine similarities between internal representations in speech recognition [96], squared ℓ_2 distance between logits [76], or KL divergence between softmax outputs [180] in adversarially robust vision models, Jensen–Shannon divergence (of three distributions) between embeddings for texture invariant image classification [62]. These are but a few highlights of the concrete and successful implementations for different applications out of a huge collection (*e.g.*, [4, 53, 132, 135, 170, 174, 186, 189]), and we can expect methods permuting these three elements (distance metrics, representation or outputs, and applications) to be invented. Further, given the popularity of GAN [48] and domain adversarial neural network [39], one can also expect the distance metric generalizes to a specialized discriminator (*i.e.* a classifier), which can be intuitively understood as a calculated (usually maximized) distance measure, Wasserstein-1 metric as an example [3, 51].

Key Differences: With this rich collection of regularizing choices, which one method should we consider in general? More importantly, do we actually need the regularization at all? These questions are important for multiple reasons, especially since sometimes consistency loss may worsen the results [74]. In this section, we first empirically show that consistency loss (especially squared ℓ_2 norm) can help learn robust and invariant models. Further, we also complement our

empirical study with theoretical analysis connecting robust error bounds with consistency loss.

There are also several previous discussions regarding the detailed understandings of data augmentation [19, 25, 45, 64, 125, 175, 182], among which, [175] is probably the most relevant as it also defends the usage of the consistency loss. In addition, our work directly connects to invariance, and shows that another advantage of consistency loss is to learn invariant representations.

3.2.3 Accuracy, Robustness, & Invariance

This section discusses the three major evaluation metrics we will use to test consistency loss. We will first recapitulate the background of accuracy and robustness, and then introduce our definition of invariance, and our proposed evaluation.

Notations $(\mathbf{X}, \mathbf{Y})$ denotes the data, where $\mathbf{X} \in \mathcal{R}^{n \times p}$ and $\mathbf{Y} \in \{0, 1\}^{n \times k}$ (one-hot vectors for k classes). $(\mathbf{x}, \mathbf{y})$ denotes a sample. $f(\cdot, \theta)$ denotes the model, which takes in the data and outputs the softmax (probabilities of the prediction) and θ denotes the corresponding parameters. $g()$ completes the prediction (*i.e.*, mapping softmax to one-hot prediction). $l(\cdot, \cdot)$ denotes a generic loss function. $a(\cdot)$ denotes a function used in the data augmentation, *i.e.*, a transformation function that alters the undesired signals of a sample. $a \in \mathcal{A}$, which is the set of transformation functions of interest. $\mathbf{P}$ denotes the distribution of $(\mathbf{x}, \mathbf{y})$. $r(\cdot; \theta)$ denotes the risk of model θ. $\widehat{\cdot}$ denotes the estimated term $\cdot$.

Accuracy

Since one of the central goals of supervised machine learning study is to improve the prediction accuracy (or to reduce the prediction error) of a model, accuracy (or error) has widely accepted definitions. For example, the community studying the statistical property of the error bound usually focuses on the expected risk defined as

$$r_{\mathbf{P}}(\widehat{\theta}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}} \mathbb{I}[g(f(\mathbf{x}; \widehat{\theta})) \neq \mathbf{y}], \tag{3.1}$$

where $\mathbb{I}[\cdot]$ is a function that returns 1 if the condition $\cdot$ holds.

In practice, the error is evaluated by replacing $\mathbf{P}$ with a hold-out test dataset, and accuracy is $1 - r_{\mathbf{P}}(\widehat{\theta})$.

Robustness

Robustness has been widely studied in the fields of cross-domain robustness [*e.g.*, 9, 114] or adversarial robustness [*e.g.*, 49, 145]. We follow the latter field and define the robustness as the worst-case expected risk when the test data is allowed to be transformed by functions in $\mathcal{A}$. Formally, we study the worst-case error as

$$r_{\mathbf{P},\mathcal{A}}(\widehat{\theta}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}} \max_{a\sim\mathcal{A}} \mathbb{I}[g(f(a(\mathbf{x}); \widehat{\theta})) \neq \mathbf{y}], \tag{3.2}$$

where we use $r_{\mathbf{P},\mathcal{A}}(\widehat{\theta})$ to denote the robust error as it will depend on $\mathcal{A}$. In practice, the robust error is also evaluated by replacing $\mathbf{P}$ with a hold-out dataset.

Invariance

While the robustness metric has been fostering the development of robust machine learning well, we notice that the metric alone may not fully reveal how the models understand the data. For example, one incentive to use data augmentation for robust models is to dilute the undesired signals in the samples, so that the models may focus more on the remaining semantic patterns in the data. While learning only semantic patterns can lead to a model excelling the robustness evaluation metrics, high robustness score does not necessarily mean the model only learns semantic patterns and discards undesired signals (as we will show later).

Invariance Metric Therefore, we argue for the need of more dedicate tests to access the model's behavior in discarding the undesired signals from data. Intuitively, if the model can learn a representation invariant to the undesired signals (that one augments data to dilute), it will map the samples of different undesired signals to the same embedding. We define the following term to measure invariance:

$$I_{\mathbf{P},\mathcal{A}}(\widehat{\theta}) = \sup_{a_1,a_2\in\mathcal{A}} D(\mathbf{Q}_{a_1(\mathbf{x}),\widehat{\theta}}, \mathbf{Q}_{a_2(\mathbf{x}),\widehat{\theta}}), \tag{3.3}$$

where $\mathbf{Q}_{a(\mathbf{x}),\widehat{\theta}}$ denotes the distribution of $f(a(\mathbf{x});\widehat{\theta})$ for $(\mathbf{x},\mathbf{y}) \sim \mathbf{P}$. $D(\cdot,\cdot)$ is a distance measure over two distributions. We suggest to use Wasserstein metric as $D(\cdot,\cdot)$, considering its favorable properties (*e.g.*, see practical examples in Figure 1 of [24] or theoretical discussions in [157]).

Invariance Test In practice, we also need to replace $\mathbf{P}$ with a hold-out dataset so that the evaluation can be performed. In addition, we notice that $I_{\mathbf{P},\mathcal{A}}(\widehat{\theta})$, although intuitive, is not convenient in practice because its values are not bounded. Thus, we reformulate it into the following invariance test procedure, whose final score will be bounded between 0 and 1 (the higher, the better), thus can be displayed in parallel with accuracy and robust accuracy.

Given a family of transformation functions used in data augmentation $\mathcal{A} = \{a_1(), a_2(), \ldots, a_t()\}$ of t elements, and a collection of samples (from the hold-out dataset) of the same label i, denoted as $\mathbf{X}^{(i)}$, the evaluation procedure is as follows. We first generate the transformed copies of $\mathbf{X}^{(i)}$ with $\mathcal{A}$, resulting in $\mathbf{X}^{(i)}_{a_1}, \mathbf{X}^{(i)}_{a_2}, \ldots, \mathbf{X}^{(i)}_{a_t}$. We combined these copies into a dataset, denoted as $\mathcal{X}^{(i)}$. For every sample $\mathbf{x}$ in $\mathcal{X}^{(i)}$, we retrieve its t nearest neighbors of other samples in $\mathcal{X}^{(i)}$, and calculate the overlap of the retrieved samples with the transformed copies of $\mathbf{x}$ by $\mathcal{A}$, *i.e.*, $\{a_1(\mathbf{x}), a_2(\mathbf{x}), \ldots, a_t(\mathbf{x})\}$. The calculated overlap score will be in $[0, 1]$ in general, but since the identify map is usually in $\mathcal{A}$, this score will usually be in $[1/t, 1]$.

During the retrieval of nearest neighbors, we consider the distance function of the two samples as $c(\cdot,\cdot) = d(f(\cdot;\widehat{\theta}) - f(\cdot;\widehat{\theta}))$, where $\widehat{\theta}$ is the model we are interested to examine. In the empirical study later, we consider $d(\cdot,\cdot) = \|\cdot,\cdot\|_1$. If we use other distance functions, the reported values may differ, but we notice that the rank of the methods compared in terms of this test barely changes.

Finally, we iterate through label i and report the averaged score for all the labels as the final reported score. A high score indicates the prediction of model $\widehat{\theta}$ is invariant to the augmentation functions in $\mathcal{A}$.

3.2.4 Empirical Study

In this section, we conduct experiments to study the relationship between robustness and invariance, as well as how training with consistency regularization can help improve the invariance score. In short, our empirical study in this section will lead us to the following three major conclusions:

- High robust accuracy does not necessarily mean a high invariance score, and vice versa.
- Consistency regularization can help improve invariance score.
- Squared ℓ_2 norm over logits seems the empirically most favorable option for learning robust and invariant representations.

Experiment Setup

Our empirical investigation is conducted over two benchmark datasets (MNIST dataset with LeNet architecture and CIFAR10 dataset with ResNet18 architecture) and three sets of the augmentations.

Augmentation Functions We consider three sets of augmentation functions:

- **Texture**: we use Fourier transform to perturb the texture of the data by discarding the high-frequency components of a radius r [166]. The smaller r is, the less high-frequency components the image has. We consider $\mathcal{A} = \{a(), a_{12}(), a_{10}(), a_8(), a_6()\}$, where the subscript denotes the radius r and $a()$ is the identity map. We consider $\mathcal{A}$ during test time, but only $a()$ and $a_6()$ during training.
- **Rotation**: we rotate the images clockwise r degrees. We consider $\mathcal{A} = \{a(), a_{15}(), a_{30}(), a_{45}(), a_{60}()\}$, where the subscript denotes the degree of rotation and $a()$ is the identity map. We consider $\mathcal{A}$ during test time, but only $a()$ and $a_{60}()$ during training.
- **Contrast**: we create the images depicting the same semantic information, but with different scales of the pixels, including the negative color representation. Therefore, we have $\mathcal{A} = \{a(\mathbf{x}) = \mathbf{x}, a_1(\mathbf{x}) = \mathbf{x}/2, a_2(\mathbf{x}) = \mathbf{x}/4, a_3(\mathbf{x}) = 1 - \mathbf{x}, a_4(\mathbf{x}) = (1 - \mathbf{x})/2, a_5(\mathbf{x}) = (1 - \mathbf{x})/4$, where $\mathbf{x}$ stands for the image whose pixel values have been set to be between 0 and 1. We consider $\mathcal{A}$ during test time, but only $a()$ and $a_3()$ during training.

Consistency Regularizations We consider the following popular choices of consistency regularization (with $\mathbf{u}$ and $\mathbf{v}$ denoting two vector embeddings):

- L: ℓ_1 norm of the vector differences, *i.e.*, $\|\mathbf{u} - \mathbf{v}\|_1$
- S: squared ℓ_2 norm of the vector differences, *i.e.*, $\|\mathbf{u} - \mathbf{v}\|_2^2$
- C: cosine similarity, *i.e.*, $\mathbf{u}^T\mathbf{v}/\|\mathbf{u}\| \cdot \|\mathbf{v}\|$
- K: KL divergence over a batch of paired embeddings; the second argument are augmented samples.
- W: Wasserstein-1 metric over a batch of paired embeddings, with implementation following Wasserstein GAN [3, 51]

D: a vanilla GAN discriminator over a batch of paired embeddings, the one-layer discriminator is trained to classify samples vs. augmented samples.

We mainly discuss applying the consistency regularization to logits (embeddings prior to the final softmax function). We have also experimented applying to the final softmax output and the embeddings one layer prior to logits. Both cases lead to substantially worse results, so we skip the discussion.

Hyperparameters We first train the baseline models to get reasonably high performance (for MNIST, we train 100 epoches with learning rate set to be 10^{-4}; for CIFAR10, we train 200 epoches with learning rate initialized to be 10^{-1} and reduced one magnitude every 50 epoches; batch sizes in both cases are set to be 128). Then we train other augmented models with the same learning rate and batch size *etc*. The regularization weight is searched with 8 choices evenly split in the logspace from 10^{-7} to 1. For each method, the reported score is from the weight resulting in highest robust accuracy. We test with three random seeds.

Evaluation Metrics: We consider the three major evaluation metrics as we discussed in §3.2.3:

A: Accuracy: test accuracy on the original test data.

R: Robustness: the worst accuracy when each sample can be transformed with $a \in \mathcal{A}$.

I: Invariance: the metric to test whether the learned representations are invariant to the augmentation functions, as introduced in §3.2.3.

Results and Discussion

Tables 3.1 and 3.2 show the empirical results across the three distribution shifts and the three evaluation metrics. First of all, no method can dominate across all these evaluations, probably because of the tradeoff between accuracy and robustness [151, 166, 180]. Similarly, tradeoff between accuracy and invariance can be expected from the role of the regularization weight: when the weight is small, the consistency regularization has no effect and the model is primarily optimized for improving accuracy; when the weight is big, the model is pushed toward a trivial solution that maps every sample to the same embedding, ignoring other patterns of the data. This is also the reason that results in Tables 3.1 and 3.2 are selected according to the robust accuracy.

Due to the tradeoffs, it may not be strategic if we only focus on the highest number of each column. Instead, we suggest to study the three rows of each test case together and compare the tradeoffs, which is also a reason we reformulate the invariance test in §3.2.3 so that we can have bounded scores directly comparable to accuracy and robust accuracy.

For example, in the texture panel (▣) of Table 3.1, while cosine similarity can outperform squared ℓ_2 norm in accuracy and robustness with a 0.3 and 0.1 margin respectively, it is disadvantageous in invariance with a larger margin (1.0). Similarly, for rotation panel (▣) of Table 3.1, KL-divergence shows the overall highest scores, followed by ℓ_1 norm and squared ℓ_2 norm. For contrast panel (▣), both ℓ_1 norm and squared ℓ_2 norm stand out. Overall, experiments in MNIST suggest the desired choice to be ℓ_1 norm or squared ℓ_2 norm, and we believe squared ℓ_2 norm is marginally better.

Table 3.1: Test results on MNIST dataset for different consistency regularizationes over three evaluation metrics and three distribution shifts. For columns: B denotes Baseline, i.e., the model does not use any data augmentation; V denotes vanilla augmentation, i.e., the model uses data augmentation but not consistency regularization; L denotes ℓ_1 norm; S denotes squared ℓ_2 norm; C denotes cosine similarity; K denotes KL divergence; W denotes Wasserstein-1 metric; D denotes GAN discriminator. For rows: ⊛ denotes texture; ⊙ denotes rotation; △ denotes contrast; A denotes accuracy; R denotes robustness; I denotes invariance.

		B	V	L	S	C	K	W	D
⊛	A	$99.2_{\pm 0.0}$	$99.2_{\pm 0.0}$	$99.0_{\pm 0.1}$	$99.1_{\pm 0.0}$	$99.4_{\pm 0.0}$	$68.7_{\pm 41}$	$98.7_{\pm 0.1}$	$99.1_{\pm 0.1}$
	R	$98.3_{\pm 0.3}$	$99.0_{\pm 0.0}$	$99.0_{\pm 0.0}$	$99.0_{\pm 0.0}$	$99.1_{\pm 0.0}$	$68.7_{\pm 41}$	$98.4_{\pm 0.1}$	$98.8_{\pm 0.1}$
	I	$92.4_{\pm 0.0}$	$99.2_{\pm 0.0}$	$100_{\pm 0.0}$	$100_{\pm 0.0}$	$99.0_{\pm 0.0}$	$76.0_{\pm 34}$	$60.7_{\pm 2.9}$	$35.0_{\pm 6.7}$
⊙	A	$99.2_{\pm 0.0}$	$99.0_{\pm 0.1}$	$99.3_{\pm 0.0}$	$99.3_{\pm 0.0}$	$99.0_{\pm 0.0}$	$98.8_{\pm 0.0}$	$98.5_{\pm 0.4}$	$98.9_{\pm 0.0}$
	R	$28.9_{\pm 0.6}$	$93.6_{\pm 0.3}$	$95.2_{\pm 0.1}$	$95.1_{\pm 0.1}$	$93.5_{\pm 0.1}$	$94.5_{\pm 0.2}$	$92.3_{\pm 0.8}$	$93.2_{\pm 0.7}$
	I	$20.6_{\pm 0.4}$	$58.3_{\pm 2.2}$	$66.0_{\pm 3.8}$	$65.4_{\pm 3.5}$	$29.1_{\pm 0.6}$	$71.9_{\pm 2.8}$	$48.7_{\pm 1.9}$	$39.3_{\pm 6.9}$
△	A	$99.2_{\pm 0.0}$	$98.9_{\pm 0.3}$	$99.4_{\pm 0.0}$	$99.4_{\pm 0.0}$	$99.2_{\pm 0.0}$	$98.9_{\pm 0.0}$	$98.7_{\pm 0.0}$	$99.1_{\pm 0.0}$
	R	$26.0_{\pm 1.0}$	$95.4_{\pm 2.6}$	$96.8_{\pm 0.8}$	$97.4_{\pm 0.6}$	$97.9_{\pm 0.4}$	$88.4_{\pm 4.5}$	$87.2_{\pm 9.6}$	$97.7_{\pm 0.6}$
	I	$20.7_{\pm 1.1}$	$37.5_{\pm 6.9}$	$41.4_{\pm 0.3}$	$41.3_{\pm 0.4}$	$26.3_{\pm 1.1}$	$40.3_{\pm 0.9}$	$28.4_{\pm 1.7}$	$20.0_{\pm 0.1}$

Table 3.2: Test results on CIFAR10 dataset for different consistency regularizationes over three evaluation metrics and three distribution shifts. Notations are the same as in Table 3.1.

		B	V	L	S	C	K	W	D
⊛	A	$88.5_{\pm 1.7}$	$86.3_{\pm 0.3}$	$82.8_{\pm 0.4}$	$82.0_{\pm 0.0}$	$86.8_{\pm 0.1}$	$84.6_{\pm 0.4}$	$86.8_{\pm 0.1}$	$86.5_{\pm 0.4}$
	R	$38.3_{\pm 0.7}$	$76.5_{\pm 0.0}$	$79.1_{\pm 0.2}$	$79.4_{\pm 0.1}$	$76.8_{\pm 0.1}$	$75.6_{\pm 0.1}$	$76.8_{\pm 0.1}$	$77.3_{\pm 0.2}$
	I	$44.7_{\pm 0.5}$	$94.1_{\pm 0.6}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	$94.2_{\pm 1.2}$	$96.7_{\pm 1.0}$	$93.4_{\pm 0.4}$	$95.4_{\pm 0.8}$
⊙	A	$88.5_{\pm 1.7}$	$81.1_{\pm 2.3}$	$80.3_{\pm 4.7}$	$78.1_{\pm 2.2}$	$80.7_{\pm 1.8}$	$80.4_{\pm 6.8}$	$87.5_{\pm 1.6}$	$83.7_{\pm 4.0}$
	R	$15.3_{\pm 1.1}$	$49.0_{\pm 0.4}$	$50.7_{\pm 2.6}$	$51.1_{\pm 1.0}$	$47.0_{\pm 0.6}$	$46.6_{\pm 4.5}$	$49.4_{\pm 0.9}$	$47.7_{\pm 1.7}$
	I	$47.3_{\pm 0.6}$	$54.6_{\pm 2.0}$	$40.7_{\pm 1.4}$	$55.2_{\pm 1.6}$	$55.7_{\pm 1.1}$	$55.5_{\pm 1.0}$	$55.7_{\pm 1.3}$	$54.5_{\pm 0.7}$
△	A	$88.5_{\pm 1.7}$	$85.2_{\pm 4.5}$	$88.8_{\pm 1.3}$	$86.9_{\pm 2.0}$	$89.6_{\pm 0.9}$	$83.4_{\pm 3.7}$	$87.2_{\pm 2.2}$	$89.7_{\pm 0.9}$
	R	$54.5_{\pm 0.8}$	$77.7_{\pm 0.8}$	$83.1_{\pm 1.3}$	$83.4_{\pm 1.1}$	$80.7_{\pm 2.4}$	$79.5_{\pm 2.9}$	$82.8_{\pm 1.0}$	$80.8_{\pm 5.6}$
	I	$53.1_{\pm 1.6}$	$67.5_{\pm 0.1}$	$69.8_{\pm 4.8}$	$71.3_{\pm 2.3}$	$53.6_{\pm 4.7}$	$73.0_{\pm 2.2}$	$74.6_{\pm 0.8}$	$66.6_{\pm 4.3}$

On the other hand, the experiments in CIFAR10 in Table 3.2 mostly favor ℓ_1 norm and squared ℓ_2 norm across the three panels, with good performances from the Wasserstein-1 metric for rotation (⊙). Also, we notice that squared ℓ_2 norm in general outperforms ℓ_1 norm.

Thus, our empirical study recommends squared ℓ_2 norm for consistency regularization to learn robust and invariant models, with ℓ_1 norm as a runner-up.

3.2.5 Analytical Support

According to our arguments in §3.2.3, Wasserstein metric is supposed to be the best option for consistency loss. However, its empirical performance does not stand out. We believe this disparity is mainly due to the difficulty in calculating the Wasserstein metric in practice.

On the other hand, norm-based consistency losses stand out. We are interested in studying the properties of these losses. In particular, our aim of this section is to complement the empirical

study by showing that squared ℓ_2 norm (or ℓ_1 norm) is a theoretically advantageous choice of consistency loss (under certain assumptions).

Also, our experiments only use two augmentation functions from $\mathcal{A}$ during training, but are tested with all the functions in $\mathcal{A}$. We also discuss the properties of these two functions and argue that training with only these two functions can be a good strategy when certain assumptions are met.

In particular, we will first formalize three properties of data augmentation functions: "dependence-preservation", "efficiency", and "vertices" (§3.2.5), and then we will show:

- When "efficiency" holds, ℓ_1 norm can replace the empirical Wasserstein metric to regularize invariance (Proposition 3.2.1 in §3.2.5).
- With the above result and "dependence-preservation", we can derive a bounded robust error if all the functions in $\mathcal{A}$ are available (Theorem 3.2.2 in §3.2.5).
- With the above result and "vertices", we can derive a bounded robust error if only two special functions in $\mathcal{A}$ are available (Lemma 3.2.3 in §3.2.5).

Well-behaved Data Transformation Function

Before we proceed to analyze the behaviors of training with data augmentations, we need first regulate some basic properties of the data transformation functions used. Intuitively, we will consider the following three properties.

A1: "Dependence-preservation" with two perspectives: Label-wise, the transformation cannot alter the label of the data, which is a central requirement of almost all the data augmentation functions in practice. Feature-wise, the transformation will not introduce new dependencies between the samples.

A2: "Efficiency": the augmentation should only generate new samples of the same label as minor perturbations of the original one. If a transformation violates this property, there should exist other simpler transformations that can generate the same target sample.

A3: "Vertices": There are extreme cases of the transformations. For example, if one needs the model to be invariant to rotations from 0° to 60°, we consider the vertices to be 0° rotation function (thus identity map) and 60° rotation function. In practice, one usually selects the transformation vertices with intuitions.

These properties are formally introduced as assumptions in Appendix A1.1. We consider **A1** an essential assumption for the rest of the analysis. We also conduct tests to see how often **A2** and **A3** can hold in practice in Appendix A1.1.

Background & Other Technical Assumptions

We first summarize a thread of previous analyses for error bounds in an extremely abstract manner. When the test data and train data are from the same distribution, many previous analyses can be sketched as:

$$r_{\mathbf{P}}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + \phi(|\Theta|, n, \delta) \tag{3.4}$$

which states that the expected risk $r_{\mathbf{P}}(\widehat{\theta})$ can be bounded by the empirical risk $\widehat{r}_{\mathbf{P}}(\widehat{\theta})$ and a function of hypothesis space $|\Theta|$ and number of samples n; δ accounts for the probability when the bound holds. $\phi()$ is a function of these three terms. Dependent on the details of different analyses, different concrete examples of this generic term will need different assumptions. We use a generic assumption **A4** to denote the assumptions required for each example (Appendix A1.1).

Following our main goal to study how consistency loss and data augmentation help in accuracy, robustness, and invariance, our strategy in theoretical analysis is to derive error bounds for accuracy and robustness, and the error bound directly contains terms to regularize the invariance. Further, as robustness naturally bounds accuracy (*i.e.*, $r_{\mathbf{P}}(\widehat{\theta}) \leqslant r_{\mathbf{P},\mathcal{A}}(\widehat{\theta})$ following the definitions in (3.1) and (3.2) respectively), we only need to study the robust error.

To study the robust error, we need two additional technical assumptions. **A5** connects the worst distribution of expected risk and the worst distribution of the empirical risk, and **A6** connects the 0-1 classification error and cross-entropy error. Details of these assumptions are in Appendix A1.1.

Regularized Worst-case Augmentation

To have a model with small invariance score, the direct approach will be regularizing the empirical counterpart of Eq. (3.3). However, Wasserstein distance is difficult to calculate in practice. Conveniently, we have the following result that links the ℓ_1 norm to the Wasserstein-1 metric in the context of data augmentation.

Proposition 3.2.1. *With A2, for any $a \in \mathcal{A}$, we have*

$$W_1(\widehat{\mathbf{Q}}_{\mathbf{x},\widehat{\theta}}, \widehat{\mathbf{Q}}_{a(\mathbf{x}),\widehat{\theta}}) = \sum_{i}^{|(\mathbf{X},\mathbf{Y})|} ||f(\mathbf{x}_i;\widehat{\theta}) - f(a(\mathbf{x}_i);\widehat{\theta})||_1,$$

where $\widehat{\mathbf{Q}}_{\mathbf{x},\widehat{\theta}}$ denotes the empirical distribution of $f(a(\mathbf{x});\widehat{\theta})$ for $(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})$.

This result conveniently allows us to use ℓ_1 norm to replace Wasserstein metric, integrating the advantages of Wasserstein metric while avoiding the practical challenges of it.

We can now offer our main technical result:

Theorem 3.2.2. *With Assumptions A1, A2, A4, A5, and A6, with probability at least $1-\delta$, we have*

$$\begin{aligned} r_{\mathbf{P},\mathcal{A}}\widehat{\theta} \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) &+ \sum_i ||f(\mathbf{x}_i;\widehat{\theta}) - f(\mathbf{x}_i';\widehat{\theta})||_1 \\ &+ \phi(|\Theta|, n, \delta) \end{aligned}$$

and $\mathbf{x}' = a(\mathbf{x})$, where $a = \arg\min_{a\in\mathcal{A}} \mathbf{y}^\top f(a(\mathbf{x});\widehat{\theta})$.

This technical result immediately inspires the method to guarantee worst case performance, as well as to explicitly enforce the concept of invariance. Notice that $a = \arg\max_{a\in\mathcal{A}} \mathbf{y}^\top f(a(\mathbf{x});\widehat{\theta})$ is simply selecting the augmentation function maximizing the cross-entropy loss, which we refer to as worst-case data augmentation. The method is also known as adversarial training [*e.g.*, 104].

Regularized Training with Vertices

Finally, as $\mathcal{A}$ in practice usually has a large number of (and possibly infinite) elements, we may not always be able to identify the worst case transformation function with reasonable computational efforts. Our final discussion is to leverage the vertex property of the transformation function to bound the worst case generalization error:

Lemma 3.2.3. *With Assumptions A1-A6, assuming there is a* $a'() \in \mathcal{A}$ *where* $\widehat{r}_{\mathbf{P}_{a'}}(\widehat{\theta}) = \frac{1}{2}\big(\widehat{r}_{\mathbf{P}_{a^+}}(\widehat{\theta}) + \widehat{r}_{\mathbf{P}_{a^-}}(\widehat{\theta})\big)$*, with probability at least* $1-\delta$*, we have:*

$$\begin{aligned} r_{\mathbf{P},\mathcal{A}}(\widehat{\theta}) \leqslant & \frac{1}{2}\big(\widehat{r}_{\mathbf{P}_{a^+}}(\widehat{\theta}) + \widehat{r}_{\mathbf{P}_{a^-}}(\widehat{\theta})\big) \\ & + \sum_i ||f(a^+(\mathbf{x}_i);\widehat{\theta}) - f(a^-(\mathbf{x}');\widehat{\theta})||_1 \\ & + \phi(|\Theta|, n, \delta), \end{aligned}$$

where $a^+()$ *and* $a^-()$ *are defined in A3.*

This result corresponds to the method that can be optimized conveniently without searching for the worst-case transformations. However, the method requires a good domain knowledge of the vertices of the transformation functions.

Thus, our theoretical discussions have supported our empirical findings in §3.2.4, with a disparity that our analytical result suggests the usage as ℓ_1 norm while our empirical study suggests the usage of squared ℓ_2 norm. We conjecture the disparity is mainly caused by the difficulty in passing the gradient of ℓ_1 norm in practice.

3.2.6 Experiments with Advanced Methods

We continue to test the methods we identified in comparison to more advanced methods. Although we argued for the value of invariance in this section, for a fair comparison, we will test the performances evaluated by the metrics the previous methods are designed for. Our method will use the same generic approach and the same augmentation functions as in previous empirical study, although these functions are not necessarily part of the distribution shift we test now. In summary, our method can outperform (or be on par with) these SOTA techniques in the robustness metric they are designed for (§3.2.6). In addition, we run a side test to show that our method can also improve accuracy (§3.2.6).

Methods

§3.2.4 and §3.2.5 lead us to test the following two methods:

- RVA (regularized vertex augmentation): using squared ℓ_2 norm as consistency regularization over logits between the original samples and the augmented samples of a fixed vertex transformation function (original samples are considered as from another vertex).
- RWA (regularized worst-case augmentation): using squared ℓ_2 norm as consistency regularization over logits between the original samples and the worst-case augmented samples identified at each iteration. Worst-case samples are the function with the maximum loss when we iterate through all the transformation functions.

Table 3.3: Comparison to advanced rotation-invariant models. We report the accuracy on the test sets rotated. "main" means the resulting images are highly likely to be semantically the same as the original ones. "all" means the average accuracy of all rotations. The underlined scores show that data augmentation and consistency regularization can help a vanilla model to compete with advanced methods. The bold scores (highest at each row) show that data augmentation and consistency regularization can further improve the advanced methods.

	ResNet			GC			ST			ETN		
	Base	RVA	RWA	Base	RVA	RWA	Base	RVA	RWA	Base	RVA	RWA
main	45.4	66.5	71.1	38.5	72.2	**73.8**	45.9	58.3	62.9	56.9	65.1	57.7
all	31.2	48.1	52.8	26.7	54.4	**55.0**	32.1	40.2	42.7	39.5	52.6	46.1

Robustness

Rotation We compare our results with rotation-invariant models, mainly Spatial Transformer (ST) [72], Group Convolution (GC) [21], and Equivariant Transformer Network (ETN) [146]. We also tried to run CGNet [81], but the method does not seem to scale to the CIFAR10 and ResNet level. All these methods are tested with ResNet34 following popular settings in the community. The results are in Table 3.3. We test the models every 15° rotation from 0° rotation to 345° rotation. Augmentation related methods are using the $\mathcal{A}$ of "rotation" in synthetic experiments, so the testing scenario goes beyond what the augmentation methods have seen during training.

We report two summary results in Table 3.3. "main" means the average prediction accuracy from images rotated from 300° to 60° (passing 0°), when the resulting images are highly likely to preserve the class label. "all" means the average accuracy of all rotations.

Our results can be interpreted with two perspectives. First, by comparing all the columns in the first panel to the first column of the other three panels, data augmentation and consistency regularization can boost a vanilla model to outperform other advanced techniques. One the other hand, by comparing the columns within each panel, data augmentation and consistency regularization can further improve the performances of these techniques.

Interestingly, the baseline model with our generic approach (RWA in the first panel) can almost compete with the advanced methods even when these methods also use augmentation and consistency regularization (RWA in GC panel). We believe this observation strongly indicates the potential of this simple augmentation and regularization method to match the advanced methods considering the margin improved by the method.

In summary, RWA can boost the vanilla model to outperform SOTA methods, and data augmentation and squared ℓ_2 consistency regularization can further improve the performances when plugged onto SOTA methods. The detailed result of each rotation are reported in Table A3 in Appendix.

Texture & Contrast We follow [5] and compare the models for a 9 super-class ImageNet classification [70] with class balanced strategies. Also, we follow [5] to report standard accuracy (Acc.), weighted accuracy (WAcc.), a scenario where samples with unusual texture are weighted more, and accuracy over ImageNet-A [61], a collection of failure cases for most Im-

Table 3.4: Comparison to advanced methods on 9 super-class ImageNet classification with different distribution shifts.

	Acc.	WAcc.	ImageNet-A	ImageNet-S
Base	90.8	88.8	24.9	41.1
SIN	88.4	86.6	24.6	40.5
LM	67.9	65.9	18.8	36.8
RUBi	90.5	88.6	27.7	42.3
RB	91.9	90.5	**29.6**	41.8
RVA	92.2	91.2	28.0	42.5
RWA	**92.8**	**91.6**	28.8	**43.2**

Table 3.5: The generic methods can also improve standard accuracy.

		Top-1	Top-5
ResNet18	Base	75.61	93.05
	RVA	76.57	93.38
	RWA	77.21	93.84
ResNet50	Base	77.39	93.96
	RVA	77.81	94.27
	RWA	78.24	94.42
ResNet101	Base	77.78	94.39
	RVA	78.18	94.51
	RWA	78.66	94.87

ageNet trained models. Additionally, we also report the performance over ImageNet-Sketch [162], an independently collected ImageNet test set with only sketch images. As [5] mainly aims to overcome the texture bias, we also use our texture-wise functions in §3.2.4 for augmentation. However, there is no direct connections between these functions and the distribution shift of the test samples. Also, we believe the distribution shifts here, especially the one introduced by our newly added ImageNet-Sketch, are more than texture, and also correspond to the contrast case of our study.

Following [5], the base network is ResNet, and we compare with the vanilla network (Base), and several methods designed for this task: including StylisedIN (SIN) [41], LearnedMixin (LM) [20], RUBi (RUBi) [16] and ReBias (RB) [5]. Results are in Table 3.4.

The results favor our generic method in most cases. RVA outperforms other methods in standard accuracy, weighted accuracy, and ImageNet-Sketch, and is shy from ReBias on ImageNet-A. RWA shows the same pattern as that of RVA, and further outperforms RVA. Overall, these results validate the empirical strength of data augmentation (even when the augmentation is not designed for the task) and squared ℓ_2 norm consistency regularization for learning robust models.

Accuracy

Further, we follow the widely-accepted CIFAR100 test pipeline and test the performances of different architectures of the ResNet family. The results are reported in Table 3.5, where Base stands for the baseline model with all default accuracy boosting techniques enabled.

For both top-1 and top-5 accuracies and across the three ResNet architectures, our techniques can help improve the accuracy. In addition, our techniques help bridge the gap of different architectures within the ResNet family: for example, RWA helps ResNet50 to outperform vanilla ResNet101.

3.2.7 Discussion

Data augmentation has benefited the development of machine learning models substantially. Given its widely usage, in this section, we seek to answer that how to train with augmented

data so that the assistance of augmentation can be taken to the fullest extent. To answer this, we first defined another dimension called invariance and conducted a line of empirical study to show that norm-based consistency loss can help learn robust and invariant models. Further, we complement our observations with formal derivations with bounded generalization errors. With progressively more specific assumptions, we identified progressively simpler methods that can bound the worst case risk. We summarize the main take-home messages below:

- Regularizing a norm distance between the logits of the originals samples and the logits of the augmented samples enjoys several merits: the trained model tend to have good worst case performance, and can learn the concept of invariance. Although our theory suggests ℓ_1 norm, but we recommend squared ℓ_2 norm in practice considering the difficulties of passing the gradient of ℓ_1 norm in backpropagation.
- With the vertex assumption held (it usually requires domain knowledge to choose the vertex functions), one can use "regularized training with vertices" (RVA) method and get good empirical performance in both accuracy and invariance, and the method is at the same complexity order of vanilla training without data augmentation. When we do not have the domain knowledge (thus are not confident in the vertex assumption), we recommend "regularized worst-case augmentation" (RWA), which has the best performance across most cases, but requires extra computations to identify the worst-case augmented samples at each iteration.

3.3 Regularization by Learning through Superficial Features

3.3.1 Background and Related Work

Imagine training an image classifier to recognize facial expressions. In the training data, while all images labeled *"smile"* may actually depict smiling people, the "smile" label might *also* be correlated with other aspects of the image. For example, people might tend to smile more often while outdoors, and to frown more in airports. In the future, we might encounter photographs with previously unseen backgrounds, and thus we prefer models that rely as little as possible on the superficial signal.

The problem of learning classifiers robust to distribution shift, commonly called *Domain Adaptation (DA)*, has a rich history. Under restrictive assumptions, such as covariate shift [50, 138], and label shift (also known as *target shift* or *prior probability shift*) [101, 134, 144, 181], principled methods exist for estimating the shifts and retraining under the importance-weighted ERM framework. Other sections bound worst-case performance under bounded shifts as measured by divergence measures on the train v.s. test distributions [9, 67, 108].

While many impossibility results for DA have been proven [10], humans nevertheless exhibit a remarkable ability to function out-of-sample, even when confronting dramatic distribution shift. Few would doubt that given photographs of smiling and frowning astronauts on the Martian plains, we could (mostly) agree upon the correct labels.

While we lack a mathematical description of how precisely humans are able to generalize so easily out-of-sample, we can often point to certain classes of perturbations that should not effect the semantics of an image. For example for many tasks, we know that the background should

not influence the predictions made about an image. Similarly, other superficial statistics of the data, such as textures or subtle coloring changes should not matter. The essential assumption of this section is that by making our model depend less on known superficial aspects, we can push the model to rely more on the *difference that makes a difference*. This section focuses on visual applications, and we focus on high-frequency textural information as the relevant notion of superficial statistics that we *do not* want our model to depend upon.

Domain generalization (DG) [114] is a variation on DA, where samples from the target domain are not available during training. In reality, data-sets may contain data cobbled together from many sources but where those sources are not labeled. For example, a common assumption used to be that there is one and only one distribution for each dataset collected, but Wang et al. [160] noticed that in video sentiment analysis, the data sources varied considerably even within the same dataset due to heterogeneous data sources and collection practices.

Domain adaptation [9, 15], and (more broadly) transfer learning have been studied for decades, with antecedents in the classic econometrics work on *sample selection bias* [58] and *choice models* [107]. For a general primer, we refer the reader to these extensive reviews [22, 169].

Domain generalization [114] is relatively new, but has also been studied extensively: covering a wide spectrum of techniques from kernel methods [34, 95, 114, 117] to more recent deep learning end-to-end methods, where the methods mostly fall into two categories: reducing the inter-domain differences of representations through adversarial (or similar) techniques [17, 44, 94, 112, 160], or building an ensemble of one-for-each-domain deep models and then fusing representations together [31, 106]. Meta-learning techniques are also explored [92]. Related studies are also conducted under the name "zero shot domain adaptation" *e.g.* [84].

3.3.2 Neural Gray-level Co-occurrence Matrix

In this section, we introduce our main technical contributions. We will first introduce the our new differentiable neural building block, NGLCM that is designed to capture textural but not semantic information from images, and then introduce our technique for excluding the textural information.

Neural Gray-Level Co-occurrence Matrix for Superficial Information Our goal is to design a neural building block that 1) has enough capacity to extract the textural information from an image, 2) is not capable of extracting semantic information. We consulted some classic computer vision techniques for inspiration and extensive experimental evidence, suggested that *gray-level co-occurrence matrix* (GLCM) [54, 86] may suit our goal. The idea of GLCM is to count the number of pixel pairs under a certain direction (common direction choices are 0 degree, 45 degree, 90 degree, and 135 degree). For example, for an image $A \in \mathcal{N}^{m \times m}$, where $\mathcal{N}$ denotes the set of all possible pixel values. The GLCM of A under the direction to 0 degree (horizontally right) will be a $|\mathcal{N}| \times |\mathcal{N}|$ matrix (denoted by G) defined as following:

$$G_{k,l} = \sum_{i=0}^{m-1} \sum_{j=0}^{m} I(A_{i,j} = k) I(A_{i+1,j} = l) \tag{3.5}$$

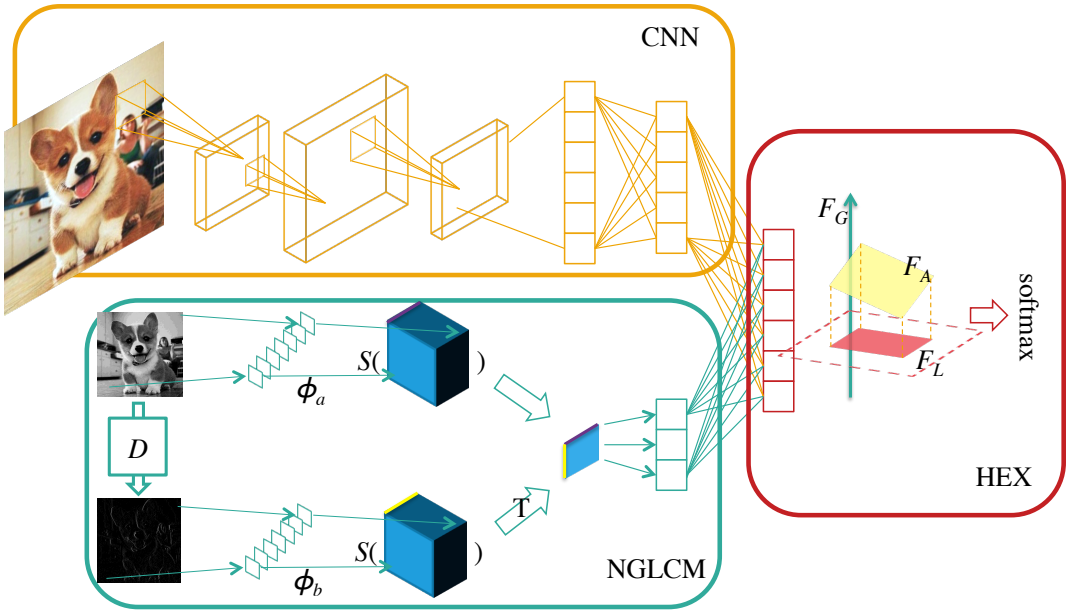


Figure 3.1: Introduction of Neural Gray-level Co-occurrence Matrix (NGLCM) and HEX.

where $|\mathcal{N}|$ stands for the cardinality of $\mathcal{N}$, $I(\cdot)$ is an identity function, i, j are indices of A, and k, l are pixel values of A as well as indices of G.

We design a new neural network building block that resembles GLCM but whose parameters are differentiable, having (sub)gradient everywhere, and thus are tunable through backpropagation.

We first flatten A into a row vector $a \in \mathcal{N}^{1\times m^2}$. The first observation we made is that the counting of pixel pairs (p_k, p_l) in Equation 3.5 is equivalent to counting the pairs $(p_k, \Delta p)$, where $\Delta p = p_k - p_l$. Therefore, we first generate a vector d by multiplying a with a matrix D, where D is designed according to the direction of GLCM. For example, D in the 0 degree case will be a $m^2 \times m^2$ matrix D such that $D_{i,i} = 1$, $D_{i,i+1} = -1$, and 0 elsewhere.

To count the elements in a and d with a differentiable operation, we introduce two sets of parameters $\phi_a \in \mathcal{R}^{|\mathcal{N}|\times 1}$ and $\phi_b \in \mathcal{R}^{|\mathcal{N}|\times 1}$ as the tunable parameter for this building block, so that:

$$G = s(a; \phi_a)s^T(d; \phi_b) \tag{3.6}$$

where $s()$ is a thresholding function defined as:

$$s(a; \phi_a) = \min(\max(a \ominus \phi_a, 0), 1)$$

where $\ominus$ denotes the minus operation with the broadcasting mechanism, yielding both $s(a; \phi_a)$ and $s(d; \phi_b)$ as $|\mathcal{N}| \times m^2$ matrices. As a result, G is a $|\mathcal{N}| \times |\mathcal{N}|$ matrix.

The design rationale is that, with an extra constrain that requires ϕ to have only unique values in the set of $\{n - \epsilon | n \in \mathcal{N}\}$, where ϵ is a small number, G in Equation 3.6 will be equivalent to the GLCM extracted with old counting techniques, subject to permutation and scale. Also, all the operations used in the construction of G have (sub)gradient and therefore all the parameters

are tunable with backpropagation. In practice, we drop the extra constraint on ϕ for simplicity in computation.

Our preliminary experiments suggested that for our purposes it is sufficient to first map standard images with 256 pixel levels to images with 16 pixel levels, which can reduce to the number of parameters of NGLCM ($|\mathcal{N}| = 16$).

HEX We first introduce notation to represent the neural network. We use $\langle X, y \rangle$ to denote a dataset of inputs X and corresponding labels y. We use $h(\cdot;\theta)$ and $f(\cdot;\xi)$ to the bottom and top components of a neural network. A conventional neural network architecture will use $f(h(X;\theta);\xi)$ to generate a corresponding result F_i and then calculate the argmax to yield the prediction label.

Besides conventional $f(h(X;\theta);\xi)$, we introduce another architecture

$$g(X;\phi) = \sigma_m((s(a;\phi_a)s^T(d;\phi_b))W_m + b_m)$$

where $\phi = \{\phi_a, \phi_b, W_m, b_m\}$, $s(a;\phi_a)s^T(d;\phi_b)$ is introduced in previous section, $\{W_m, b_m, \sigma_m\}$ (weights, biases, and activation function) form a standard MLP.

With the introduction of $g(\cdot;\phi)$, the final classification layer turns into $f[h(X;\theta), g(X;\phi)];\xi)$ (where we use $[\cdot,\cdot]$ to denote concatenation).

Now, with the representation learned through raw data by $h(\cdot;\theta)$ and textural representation learned by $g(\cdot;\phi)$, the next question is to force $f(\cdot;\xi)$ to predict with transformed representation from $h(\cdot;\theta)$ that *in some sense* independent of the superficial representation captured by $g(\cdot;\phi)$.

To illustrate following ideas, we first introduce three different outputs from the final layer:

$$\begin{aligned} F_A &= f([h(X;\theta), g(X;\phi)];\xi) \\ F_G &= f([\mathbf{0}, g(X;\phi)];\xi) \\ F_P &= f([h(X;\theta), \mathbf{0}];\xi) \end{aligned} \tag{3.7}$$

where F_A, F_G, and F_P stands for the results from both representations (concatenated), only the textural information (prepended with the 0 vector), and only the raw data (concatenated wit hthe 0 vecotr), respectively. $\mathbf{0}$ stands for a padding matrix with all the zeros, whose shape can be inferred by context.

Several heuristics have been proposed to force a network to "forget" some part of a representation, such as adversarial training [39] or information-theoretic regularization [113]. In similar spirit, our first proposed solution is to adopt the reverse gradient idea [39] to train F_P to be predictive for the semantic labels y while forcing the F_P to be invariant to F_G. Later, we refer to this method as *ADV*. When we use a multilayer perceptron (MLP) to try to predict $g(X;\phi)$ from $h(X;\theta)$ and update the primary model to *fool* the MLP via reverse gradient, we refer to the model as *ADVE*.

Additionally, we introduce a simple alternative. Our idea lies in the fact that, in an affine space, to find a transformation of representation A that is least explainable by some other representation B, a straightforward method will be projecting A with a projection matrix constructed by B (sometimes referred as residual maker matrix.). To utilize this linear property, we choose to work on the space of F generated by $f(\cdot;\xi)$ right before the final argmax function.

Projecting F_A with

$$F_L = (I - F_G(F_G^T F_G)^{-1} F_G^T) F_A \tag{3.8}$$

will yield F_L for parameter tuning. All the parameters ξ, ϕ, θ can be trained simultaneously. In testing time, F_P is used.

Two alternative forms of our algorithm are also worth mentioning: 1) During training, one can tune an extra hyperparameter (λ) through

$$l(\arg\max F_L, y) + \lambda l(\arg\max F_G, y)$$

to ensure that the NGLCM component is learning superficial representations that are related to the present task where $l(\cdot, \cdot)$ is a generic loss function. 2) During testing, one can use F_L, although this requires evaluating the NGLCM component at prediction time and thus is slightly slower. We experimented with these three forms with our synthetic datasets and did not observe significant differences in performance and thus we adopt the fastest method as the main one.

Empirically, we also notice that it is helpful to make sure the textural representation $g(X; \phi)$ and raw data representation $h(X; \theta)$ are of the same scale for HEX to work, so we column-wise normalize these two representations in every minibatch.

Experiments To show the effectiveness of our proposed method, we conduct range of experiments, evaluating HEX's resilience against dataset shift. To form intuition, we first examine the NGLCM and HEX separately with two basic testings, then we evaluate on two synthetic datasets, on in which *dataset shift* is introduced at the semantic level and another at the raw feature level, respectively. We finally evaluate other two standard domain generalization datasets to compare with the state-of-the-art. All these models are trained with ADAM [79].

We conducted ablation tests on our two synthetic datasets with two cases 1) replacing NGLCM with one-layer MLP (denoted as **M**), 2) not using HEX/ADV (training the network with F_A (Equation 3.7) instead of F_L (Equation 3.8)) (denoted as **N**). We also experimented with the two alternative forms of HEX: 1) with F_G in the loss and $\lambda = 1$ (referred as HEX-ADV), 2) predicting with F_L (referred as HEX-ALL). We also compare with the popular DG methods (DANN [39]) and another method called information-dropout [1].

NGLCM Only Extracts Textural Information To show that the NGLCM only extracts textural information, we trained the network with a mixture of four digit recognition data sets: MNIST [88], SVHN [116], MNIST-M [38], and USPS [28]. We compared NGLCM with a single layer of MLP. The parameters are trained to minimize prediction risk of *digits* (instead of *domain*). We extracted the representations of NGLCM and MLP and used these representations as features to test the five-fold cross-validated Naïve Bayes classifier's accuracy of predicting digit and domain. With two choices of learning rates, we repeated this for every epoch through 100 epochs of training and reported the mean and standard deviation over 100 epochs in Table 3.6: while MLP and NGLCM perform comparably well in extracting textural information, NGLCM is significantly less useful for recognizing the semantic label.

HEX projection To test the effectiveness of HEX, we used the extracted SURF [7] features (800 dimension) and GLCM [86] features (256 dimension) from office data set [130] (31 classes). We built a two-layer MLP (800×256, and 256×31) as baseline that only predicts with SURF

	Random	MLP (1e-2)	NGLCM (1e-2)	MLP (1e-4)	NGLCM (1e-4)
Domain	0.25	0.686±0.020	0.738±0.018	0.750±0.054	0.687±0.029
Label	0.1	0.447±0.039	0.161±0.008	0.534±0.022	0.142±0.023

Table 3.6: Accuracy of domain classification and digit classification

Train	Test	Baseline	HEX	HEX-ADV	HEX-ALL
A, W	D	0.405±0.016	0.343±0.030	0.343±0.030	0.216±0.119
D, W	A	0.112±0.008	0.147±0.004	0.147±0.004	0.055±0.004
A, D	W	0.400±0.016	0.378±0.034	0.378±0.034	0.151±0.008

Table 3.7: Accuracy on Office data set with extracted features. The Baseline refers to MLP with SURF features. The HEX methods refer to adding another MLP with features extracted by traditional GLCM methods. Because D and W are similar domains (same obejcts even share the same background), we believe these results favor the HEX method (see Section 3.3.2) for duscussion).

features. This architecture and corresponding learning rate are picked to make sure the baseline can converge to a relatively high prediction performance. Then we plugged in the GLCM part with an extra first-layer network 256×32 and the second layer of the baseline is extended to 288×31 to take in the information from GLCM. Then we train the network again with HEX with the same learning rate.

The Office data set has three different subsets: Webcam (W), Amazon (A), and DSLR (D). We trained and validated the model on a mixture of two and tested on the third one. We ran five experiments and reported the averaged accuracy with standard deviation in Table 3.7. These performances are not comparable to the state-of-the-art because they are based on features. At first glance, one may frown upon on the performance of HEX because out of three configurations, HEX only outperforms the baseline in the setting $\{W, D\} \rightarrow A$. However, a closer look into the datasets gives some promising indications for HEX: we notice W and D are distributed similarly in the sense that objects have similar backgrounds, while A is distributed distinctly. Therefore, if we assume that there are two classifiers C_1 and C_2: C_1 can classify objects based on object feature and background feature while C_2 can only classify objects based on object feature ignoring background feature. C_2 will only perform better than C_1 in $\{W, D\} \rightarrow A$ case, and will perform worse than C_2 in the other two cases, which is exactly what we observe with HEX.

Facial Expression Classification with Nuisance Background We generated a synthetic data set extending the Facial Expression Research Group Database [2], which is a dataset of six animated individuals expressing seven different sentiments. For each pair of individual and sentiment, there are over 1000 images. To introduce the data shift, we attach seven different backgrounds to these images. In the training set (50% of the data) and validation set (30% of the data), the background is correlated with the sentiment label with a correlation of ρ; in testing set (the rest 20% of the data), the background is independent of the sentiment label. In the experiment, we format the resulting images to 28×28 grayscale images.

We run the experiments first with the baseline CNN (two convolutional layers and two fully

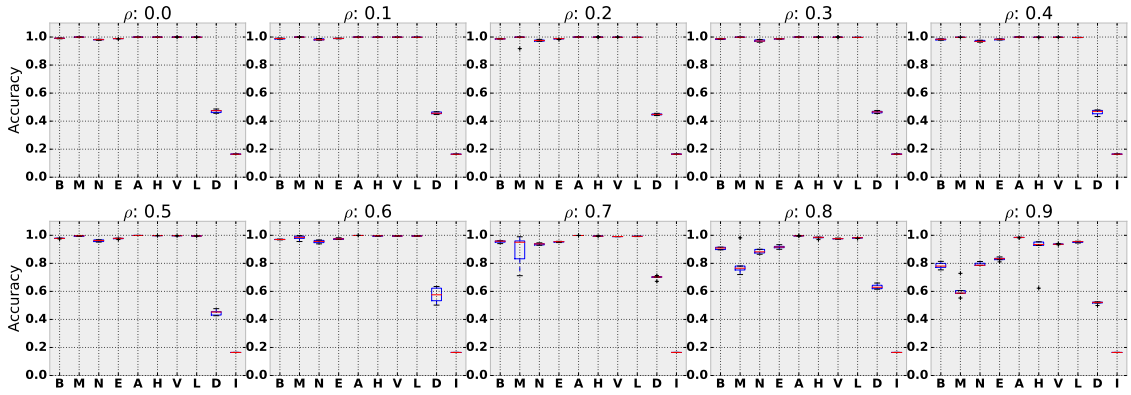


Figure 3.2: Averaged testing accuracy and standard deviation of five repeated experiments with different correlation level on sentiment with nuisance background data. Notations: baseline CNN (**B**), Ablation Tests (**M** (replacing NGLCM with MLP) and **N** (training without HEX projection)), ADVE (**E**), ADV (**A**), HEX (**H**), HEX-ADV (**V**), HEX-ALL (**L**), DANN (**G**), and InfoDropout (**I**).

connected layers) to tune for hyperparameters. We chose to run 100 epochs with learning rate 5e-4 because this is when the CNN can converge for all these 10 synthetic datasets. We then tested other methods with the same learning rate. The results are shown in Figure 3.2 with testing accuracy and standard deviation from five repeated experiments. Testing accuracy is reported by the model with the highest validation score. In the figure, we compare baseline CNN (**B**), Ablation Tests (**M** and **N**), ADV (**A**), HEX (**H**), DANN (**G**), and InfoDropout (**I**). Most these methods perform well when ρ is small (when testing distributions are relatively similar to training distribution). As ρ increases, most methods' performances decrease, but Adv and HEX behave relatively stable across these ten correlation settings. We also notice that, as the correlation becomes stronger, **M** deteriorates at a faster pace than other methods. Intuitively, we believe this is because the MLP learns both from the *semantic* signal together with superficial signal, leading to inferior performance when HEX projects this signal out. We also notice that ADV and HEX improve the speed of convergence.

Mitigating the Tendency of Surface Statistical Regularities in MNIST As [75] observed, CNNs have a tendency to learn the surface statistical regularities: the generalization of CNNs is partially due to the abstraction of high level semantics of an image, and partially due to surface statistical regularities. Here, we demonstrate the ability of HEX to overcome such tendencies. We followed the radial and random Fourier filtering introduced in [75] to attach the surface statistical regularities into the images in MNIST. There are three different regularities altogether (radial kernel, random kernel, and original image). We attached two of these into training and validation images and the remaining one into testing images. We also adopted two strategies in attaching surface patterns to training/validation images: 1) *independently*: the pattern is independent of the digit, and 2) *dependently*: images of digit 0-4 have one pattern while images of digit 5-9 have the other pattern.

We used the same learning rate scheduling strategy as in the previous experiment. The results

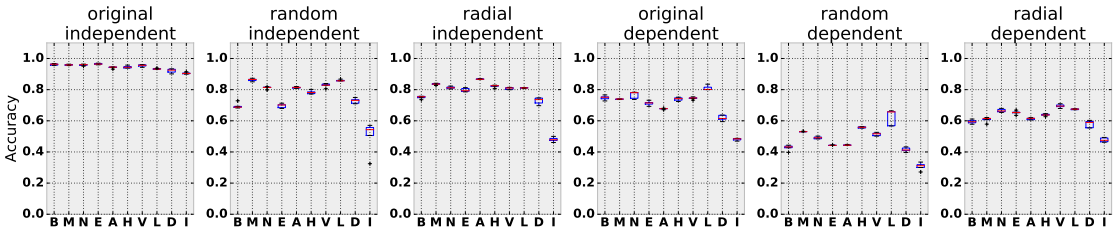


Figure 3.3: Averaged testing accuracy and standard deviation of five repeated experiments with different strategies of attaching patterns to MNIST data. Notations: baseline CNN (**B**), Ablation Tests (**M** (replacing NGLCM with MLP) and **N** (training without HEX projection)), ADVE (**E**), ADV (**A**), HEX (**H**), HEX-ADV (**V**), HEX-ALL (**L**), DANN (**G**), and InfoDropout (**I**).

Test	CAE	MTAE	CCSA	DANN	Fusion	LabelGrad	CrossGrad	HEX	ADV
$\mathcal{M}_{0degree}$	72.1	82.5	84.6	86.7	85.6	89.7	88.3	**90.1**	**89.9**
$\mathcal{M}_{15}$	95.3	96.3	95.6	98	95.0	97.8	98.6	**98.9**	98.6
$\mathcal{M}_{30}$	92.6	93.4	94.6	97.8	95.6	98.0	98.0	**98.9**	98.8
$\mathcal{M}_{45}$	81.5	78.6	82.9	97.4	95.5	97.1	97.7	**98.8**	98.7
$\mathcal{M}_{60}$	92.7	94.2	94.8	96.9	95.9	96.6	97.7	98.3	**98.6**
$\mathcal{M}_{75}$	79.3	80.5	82.1	89.1	84.3	**92.1**	91.4	90.0	90.4
Avg	85.6	87.6	89.1	94.3	92.0	95.2	95.3	**95.8**	95.2

Table 3.8: Accuracy on MNIST-Rotation data set

are shown in Figure 3.3. Figure legends are the same as previous. Interestingly, NGLCM and HEX contribute differently across these cases. When the patterns are attached independently, **M** performs the best overall, but when the patterns are attached dependently, **N** and HEX perform the best overall. In the most challenging case of these experiments (random kerneled as testing, pattern attached dependently), HEX shows a clear advantage. Also, HEX behaves relatively more stable overall.

MNIST with Rotation as Domain We continue to compare HEX with other state-of-the-art DG methods (that use distribution labels) on popular DG data sets. We experimented with the MNIST-rotation data set, on which many DG methods have been tested. The images are rotated with different degrees to create different domains. We followed the approach introduced by [44]. To reiterate: we randomly sampled a set $\mathcal{M}$ of 1000 images out of MNIST (100 for each label). Then we rotated the images in $\mathcal{M}$ counter-clockwise with different degrees to create data in other domains, denoted by $\mathcal{M}_{15}$, $\mathcal{M}_{30}$, $\mathcal{M}_{45}$, $\mathcal{M}_{60}$, $\mathcal{M}_{75}$. With the original set, denoted by $\mathcal{M}_{0degree}$, there are six domains altogether.

We compared the performance of HEX/ADV with several methods tested on this data including CAE [128], MTAE [44], CCSA [112], DANN [39], Fusion [106], LabelGrad, and CrossGrad [136]. The results are shown in Table 3.8: HEX is only inferior to previous methods in one case and leads the average performance overall.

PACS: Generalization in Photo, Art, Cartoon, and Sketch Finally, we tested on the PACS data set [91], which consists of collections of images of seven different objects over four domains,

Test Domain	AlexNet	DSN	L-CNN	MLDG	Fusion	HEX	ADV
Art	63.3	61.1	62.8	63.6	64.1	**66.8**	64.9
Cartoon	63.1	66.5	66.9	63.4	66.8	**69.7**	69.6
Photo	87.7	83.2	89.5	87.8	**90.2**	87.9	88.2
Sketch	54	58.5	57.5	54.9	**60.1**	56.3	55.5
Average	67.0	67.3	69.2	67.4	**70.3**	70.2	69.5

Table 3.9: Testing Accuracy on PACS

including photo, art painting, cartoon, and sketch.

Following [91], we used AlexNet as baseline method and built HEX upon it. We met some optimization difficulties in directly training AlexNet on PACS data set with HEX, so we used a heuristic training approach: we first fine-tuned the AlexNet pretrained on ImageNet with PACS data of training domains without plugging in NGLCM and HEX, then we used HEX and NGLCM to further train the top classifier of AlexNet while the weights of the bottom layer are fixed. Our heuristic training procedure allows us to tune the AlexNet with only 10 epoches and train the top-layer classifier 100 epochs (roughly only 600 seconds on our server for each testing case).

We compared HEX/ADV with the following methods that have been tested on PACS: AlexNet (directly fine-tuning pretrained AlexNet on PACS training data [91]), DSN [12], L-CNN [91], MLDG [92], Fusion [106]. Notice that most of the competing methods (DSN, L-CNN, MLDG, and Fusion) have explicit knowledge about the domain identification of the training images. The results are shown in Table 3.9. Impressively, HEX is only slightly shy of Fusion in terms of overall performance. Fusion is a method that involves three different AlexNets, one for each training domain, and a fusion layer to combine the representation for prediction. The Fusion model is roughly three times bigger than HEX since the extra NGLCM component used by HEX is negligible in comparison to AlexNet in terms of model complexity. Interestingly, HEX achieves impressively high performance when the testing domain is Art painting and Cartoon, while Fusion is good at prediction for Photo and Sketch.

Discussion We introduced two novel components: NGLCM that only extracts textural information from an image, and HEX that projects the textural information out and forces the model to focus on semantic information. Limitations still exist. For example, NGLCM cannot be completely free of semantic information of an image. As a result, if we apply our method on standard MNIST data set, we will see slight drop of performance because NGLCM also learns some semantic information, which is then projected out. Also, training all the model parameters simultaneously may lead into a trivial solution where F_G (in Equation 3.7) learns garbage information and HEX degenerates to the baseline model. To overcome these limitations, we invented several training heuristics, such as optimizing F_P and F_G sequentially and then fix some weights. However, we did not report results with training heuristics (expect for PACS experiment) because we hope to simplify the methods. Another limitation we observe is that sometimes the training performance of HEX fluctuates dramatically during training, but fortunately, the model picked up by highest validation accuracy generally performs better than competing methods. Despite

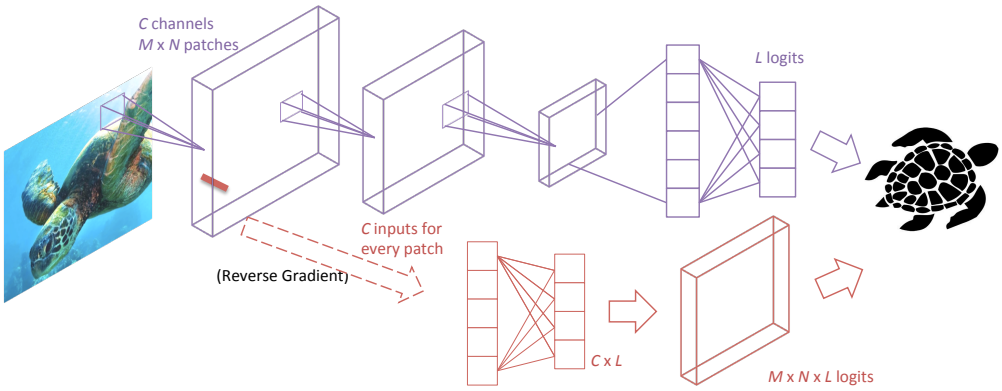


Figure 3.4: In addition to the primary classifier, our model consists of a number of side classifiers, applied at each 1×1 location in a designated early layer. The side classifiers result in one prediction per spatial location. The goal of **patch-wise adversarial regularization** is to fool all of them (via reverse gradient) while nevertheless outputting the correct class from the topmost layer.

these limitations, we still achieved impressive performance on both synthetic and popular DG data sets.

3.3.3 Patch-wise Adversarial Regularization

We use $\langle \mathbf{X}, \mathbf{y} \rangle$ to denote the samples and $f(g(\cdot;\delta);\theta)$ to denote a convolutional neural network, where $g(\cdot;\delta)$ denotes the output of the bottom convolutional layers (e.g., the first layer), and δ and θ are parameters to be learned. The traditional training process addresses the optimization problem

$$\min_{\delta,\theta} \mathbb{E}_{(\mathbf{X},\mathbf{y})}[l(f(g(\mathbf{X};\delta);\theta),\mathbf{y})], \tag{3.9}$$

where $l(\cdot,\cdot)$ denotes the loss function, commonly cross-entropy loss in classification problems.

Following the standard set-up of a convolutional layer, δ is a tensor of $c \times m \times n$ parameters, where c denotes the number of convolutional channels, and $m \times n$ is the size of the convolutional kernel. Therefore, for the i^{th} sample, $g(\mathbf{X}_i;\delta)$ is a representation of $\mathbf{X}_i$ of the dimension $c \times m' \times n'$, where m' (or n') is a function of the image dimension and m (or n). [1]

Patch-wise Adversarial Regularization We first introduce a new classifier, $h(\cdot;\phi)$ that takes the input of a c-length vector and predicts the label. Thus, $h(\cdot;\phi)$ can be applied onto the representation $g(\mathbf{X}_i;\delta)$ and yield $m' \times n'$ predictions. Therefore, each of the $m' \times n'$ predictions can be seen as a prediction made only by considering a small image patch corresponding to each of the receptive fields in $g(\mathbf{X}_i;\delta)$. If any of the image patches are predictive and $g(\cdot;\delta)$ summarizes the predictive representation well, $h(\cdot;\phi)$ can be trained to achieve a high prediction accuracy.

[1] The exact function depends on padding size and stride size, and is irrelevant to the discussion of this section.

On the other hand, if $g(\cdot;\delta)$ summarizes the patch-wise predictive representation well, higher layers ($f(\cdot;\theta)$) can directly utilize these representation for prediction and thus may not be required to learn a global concept. Our intuition is that by regularizing $g(\cdot;\delta)$ such that each fiber (i.e., representation at the same location from every channel) in the activation tensor should not be individually predictive of the label, we can prevent our model from relying on local patterns and instead force it to learn a pattern that can only be revealed by aggregating information across multiple receptive fields.

As a result, in addition to the standard optimization problem (Eq. 3.9), we also optimize the following term:

$$\min_{\phi}\max_{\delta}\mathbb{E}_{(\mathbf{X},\mathbf{y})}[\sum_{i,j}^{m',n'} l(h(g(\mathbf{X};\delta)_{i,j};\phi),\mathbf{y})] \tag{3.10}$$

where the minimization consists of training $h(\cdot;\phi)$ to predict the label based on the local features (at each spatial location) while the maximization consists of training $g(\cdot;\delta)$ to shift focus away from local predictive representations.

We hypothesize that by jointly solving these two optimization problems (Eq. 3.9 and Eq. 3.10), we can train a model that can predict the label well without relying too strongly on local patterns. The optimization can be reformulated into the following two problems:

$$\min_{\delta,\theta}\mathbb{E}_{(\mathbf{X},\mathbf{y})}[l(f(g(\mathbf{X};\delta);\theta),\mathbf{y}) - \frac{\lambda}{m'n'}\sum_{i,j}^{m',n'} l(h(g(\mathbf{X};\delta)_{i,j};\phi),\mathbf{y})]$$

$$\min_{\phi}\mathbb{E}_{(\mathbf{X},\mathbf{y})}[\frac{\lambda}{m'n'}\sum_{i,j}^{m',n'} l(h(g(\mathbf{X};\delta)_{i,j};\phi),\mathbf{y})]$$

where λ is a tuning hyperparameter. We divide the loss by $m'n'$ to keep the two terms at a same scale.

Our method can be implemented efficiently as follows: In practice, we consider $h(\cdot;\phi)$ as a fully-connected layer. ϕ consists of a $c \times k$ weight matrix and a k-length bias vector, where k is the number of classes. The $m' \times n'$ forward operations as fully-connected networks can be efficiently implemented as a 1×1 convolutional operation with c input channels and k output channels operating on the $m' \times n'$ representation.

Note that although the input has $m' \times n'$ vectors, $h(\cdot;\phi)$ only has one set of parameters that is used for all these vectors, in contrast to building a set of parameter for every receptive field of the $m' \times n'$ dimension. Using only one set of parameters can not only help to reduce the computational load and parameter space, but also help to identify the predictive local patterns well because the predictive local pattern does not necessarily appear at the same position across the images. Our idea of our method is illustrated in Figure 3.4.

Other Extensions and Training Heuristics There can be many simple extensions to the basic PAR setting we discussed above. Here we introduce three extensions that we will experiment with later in the experiment section.

More Powerful Pattern Classifier: We explore the space of discriminator architectures, replacing the single-layer network $h(\cdot;\phi)$ with a more powerful network architecture, e.g. a multilayer perceptron (MLP). In this section, we consider three-layer MLPs with ReLU activation functions. We name this variant as PAR_M.

Broader Local Pattern: We can also extend the 1×1 convolution operation to enlarge the concept of "local". In this section, we experiment with a 3×3 convolution operation, thus the number of parameters in ϕ is increased. We refer to this variant as PAR_B.

Higher Level of Local Concept: Further, we can also build the regularization upon higher convolutional layers. Building the regularization on higher layers is related to enlarging the patch of image, but also considering higher level of abstractions. In this section, we experiment the regularization on the second layer. We refer this method as PAR_H.

Training Heuristics: Finally, we introduce the training heuristic that plays an important role in our regularization technique, especially in modern architectures such as AlexNet or ResNet. The training heuristic is simple: we first train the model conventionally until convergence (or after a certain number of epochs), then train the model with our regularization. In other words, we can also directly work on pretrained models and continue to fine-tune the parameters with our regularization.

Experiments In this section, we test PAR over a variety of settings, we first test with perturbed MNIST under the domain generalization setting, and then test with perturbed CIFAR10 under domain adaptation setting. Further, we test on more challenging data sets, with PACS data under domain generalization setting and our newly proposed ImageNet-Sketch data set. We compare with previous state-of-the-art when available, or with the most popular benchmarks such as DANN [39], InfoDrop [1], and HEX [163] on synthetic experiments.[2,3]

MNIST with Perturbation We follow the set-up of Wang et al. [163] in experimenting with MNIST data set with different superficial patterns. There are three different superficial patterns (radial kernel, random kernel, and original image). The training/validation samples are attached with two of these patterns, while the testing samples are attached with the remaining one. As in Wang et al. [163], training/validation samples are attached with patterns following two strategies: 1) *independently*: the pattern is independent of the digit, and 2) *dependently*: images of digit 0-4 have one pattern while images of digit 5-9 have the other pattern.

We use the same model architecture and learning rate as in Wang et al. [163]. The extra hyperparameter λ is set as 1 as the most straightforward choice. Methods in Wang et al. [163] are trained for 100 epochs, so we train the model for 50 epochs as pretraining and 50 epochs with our regularization. The results are shown in Figure 3.5. In addition to the direct message that our proposed method outperforms competing ones in most cases, it is worth mentioning that the proposed methods behave differently in the "dependent" settings. For example, PAR_M performs the best in the "original" and "radial" settings, but almost the worst among proposed methods in the "random" setting, which may indicate that the pattern attached by "random" kernel can be more easily detected and removed by PAR_M during training (Notice that the name of the setting ("original", "radial" or "random") indicates the pattern attached to testing images, and

[2]Clean demonstration of the implementation can be found at: https://github.com/HaohanWang/PAR

[3]Source code for replication can be found at : https://github.com/HaohanWang/PAR_experiments

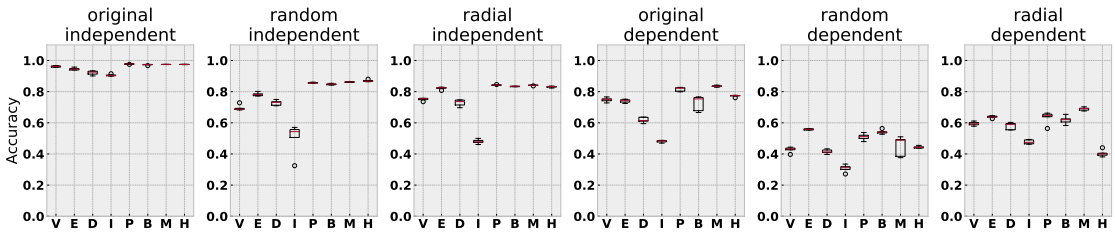


Figure 3.5: Prediction accuracy with standard deviation for MNIST with patterns. Notations: V: vanilla baseline, E: HEX, D: DANN, I: InfoDrop, P: PAR, B: PAR_B, M: PAR_M, H: PAR_H

Table 3.10: Test accuracy of PAR and variants on Cifar10 datasets with perturbed color and texture.

	ResNet	DANN	InfoDrop	HEX	PAR	PAR_B	PAR_M	PAR_H
Greyscale	87.7	87.3	86.4	87.6	**88.1**	87.9	87.8	86.9
NegColor	62.8	64.3	57.6	62.4	66.2	65.3	**67.6**	62.7
RandKernel	43.0	33.4	41.3	42.5	47.0	40.5	**47.5**	40.8
RadialKernel	62.4	63.3	60.3	61.9	**63.8**	63.2	63.2	61.4
Average	63.9	62.0	61.4	63.6	66.3	64.2	**66.5**	62.9

the training samples are attached with the other two patterns).

CIFAR with Perturbation We continue to experiment on CIFAR10 data set by modifying the color and texture of test dataset with four different schemas: 1) greyscale; 2) negative color; 3) random kernel; 4) radial kernel. Some examples of the perturbed data are shown in Appendix. In this experiment, we use ResNet-32 as our base classifier, which has a rough 92% prediction accuracy on original CIFAR10 test data set.

As for PAR, we first train the base classifier for 250 epochs and then train with the adversarial loss for another 150 epochs. As for the competing models, we also train for 400 epochs with carefully selected hyperparameters. The overall performances are shown in Table 3.10. In general, PAR and its variants achieve the best performances on all four test data sets, even when DANN has an unfair advantage over others by seeing unlabelled testing data during training. To be specific, PAR achieves the best performances on the greyscale and radial kernel settings; PAR_M is the best on the negative color and random kernel settings. One may argue that the numeric improvements are not significant and PAR may only affect the model marginally, but a closer look at the training process of the methods indicates that our regularization of local patterns benefits the robustness significantly while minimally impacting the original performance. More detailed discussions are in Appendix.

PACS We test on the PACS data set [91], which consists of collections of images over four domains, including photo, art painting, cartoon, and sketch. Many recent methods have been tested on this data set, which offers a convenient way for PAR to be compared with the previous state-of-the-art. Following Li et al. [91], we use AlexNet as baseline and build PAR upon it. We compare with recently reported state-of-the-art on this data set, including DSN [12], LCNN [91], MLDG [92], Fusion [106], MetaReg [6], Jigen [18], and HEX [163], in addition to the baseline

reported in [91]. We are also aware that methods that explicitly use domain knowledge [*e.g.*, 89] may be helpful, but we do not directly compare with them numerically, as the methods deviate from the central theme of this section.

Table 3.11: Prediction accuracy of PAR and variants on PACS data set in comparison with the previously reported state-of-the-art results. Bold numbers indicate the best performance (three sets, one for each scenario). We use * to denote the methods that use the training setting in [18] (*e.g.*, extra data augmentation, different train-test split, and different learning rate scheduling). Notably, PAR_H achieves the best performance in sketch testing case even in comparison to all other methods without data augmentation.

	Art	Cartoon	Photo	Sketch	Average	Forgoing Domain ID	Data Aug.
AlexNet	63.3	63.1	87.7	54	67.03	Y	
DSN	61.1	66.5	83.2	58.5	67.33		
L-CNN	62.8	66.9	89.5	57.5	69.18		
MLDG	63.6	63.4	87.8	54.9	67.43		
Fusion	64.1	66.8	90.2	**60.1**	70.30		
MetaReg	**69.8**	**70.4**	**91.1**	59.2	**72.63**		
HEX	66.8	69.7	87.9	56.3	70.18	Y	
PAR	**66.9**	67.1	88.6	62.6	71.30	Y	
PAR_B	66.3	67.8	87.2	61.8	70.78	Y	
PAR_M	65.7	68.1	88.9	61.7	71.10	Y	
PAR_H	66.3	**68.3**	**89.6**	**64.1**	**72.08**	Y	
Jigen*	67.6	**71.7**	89.0	**65.1**	73.38	Y	Y
PAR*	68.0	71.6	**90.8**	61.8	73.05	Y	Y
$\text{PAR}_\text{B}^\star$	67.6	70.7	90.1	62.0	72.59	Y	Y
$\text{PAR}_\text{M}^\star$	**68.7**	71.5	90.5	62.6	73.33	Y	Y
$\text{PAR}_\text{H}^\star$	**68.7**	70.5	90.4	64.6	***73.54***	Y	Y

Following the training heuristics we introduced, we continue with trained AlexNet weights[4] and fine-tune on training domain data of PACS for 100 epochs. We notice that once our regularization is plugged in, we can outperform the baseline AlexNet with a 2% improvement. The results are reported in Table 3.11, where we separate the results of techniques relying on domain identifications and techniques free of domain identifications.

We also report the results based on the training schedule used by [18] as shown in the bottom part of Table 3.11. Note that [18] used the random training-test split that are different from the official split used by the other baselines. In addition, they used another data augmentation technique to convert image patch to grayscale which could benefit the adaptation to Sketch domain.

While our methods are in general competitive, it is worth mentioning that our methods improve upon previous methods with a relatively large margin when Sketch is the testing domain. The improvement on Sketch is notable because Sketch is the only colorless domain out of the four domains in PACS. Therefore, when tested with the other three domains, a model may learn

[4]https://www.cs.toronto.edu/~guerzhoy/tf_alexnet/

Table 3.12: Testing accuracy of competing methods on the ImageNet-Sketch data. The bottom half denotes the method that has extra advantages: † denotes the method that has access to unlabelled target domain data, and * denotes the method that use extra data augmentation.

	AlexNet	InfoDrop	HEX	PAR	PAR_{B}	PAR_{M}	PAR_{H}
Top 1	0.1204	0.1224	0.1292	**0.1306**	0.1273	0.1287	0.1266
Top 5	0.2480	0.2560	**0.2654**	0.2627	0.2575	0.2603	0.2544
		DANN†	JigGen*	PAR*	$\text{PAR}_{\text{B}}{}^{\star}$	$\text{PAR}_{\text{M}}{}^{\star}$	$\text{PAR}_{\text{H}}{}^{\star}$
Top 1		0.1360	0.1469	0.1494	0.1494	**0.1501**	0.1499
Top 5		0.2712	0.2898	0.2949	0.2945	**0.2957**	0.2954

to exploit the color information, which is usually local, to predict, but when tested with Sketch domain, the model has to learn colorless concepts to make good predictions.

ImageNet-Sketch We use AlexNet as the baseline and test whether our method can help improve the out-of-domain prediction. We start with ImageNet pretrained AlexNet and continue to use PAR to tune AlexNet for another five epochs on the original ImageNet training dataset. The results are reported in Table 3.12.

We are particularly interested in how PAR improves upon AlexNet, so we further investigate the top-1 prediction results. Although the numeric results in Table 3.12 seem to show that PAR only improves the upon AlexNet by predicting a few more examples correctly, we notice that these models share 5025 correct predictions, while AlexNet predicts another 1098 images correctly and PAR predicts a different set of 1617 images correctly.

We first investigate the examples that are correctly predicted by the original AlexNet, but wrongly predicted by PAR. We notice some examples that help verify the performance of PAR. For examples, PAR incorrectly predicts three instances of "keyboard" as "crossword puzzle," while AlexNet predicts these samples correctly. It is notable that two of these samples are "keyboards with missing keys" and hence look similar to a "crossword puzzle."

We also investigate the examples that are correctly predicted by PAR, but wrongly predicted by the original AlexNet. Interestingly, we notice several samples that are wrongly predicted by AlexNet because the model may only focus on the local patterns. The first example is a stethoscope, PAR predicts it correctly with 0.66 confidence, while AlexNet predicts it to be a hook. We conjecture the reason to be that AlexNet tends to only focus on the curvature which resembles a hook. The second example tells a similar story, PAR predicts tricycle correctly with 0.92 confidence, but AlexNet predicts it as a safety pin with 0.51 confidence. We believe this is because part of the image (likely the seat-supporting frame) resembles the structure of a safety pin. For the third example, PAR correctly predicts it to be an Afghan hound with 0.89 confidence, but AlexNet predicts it as a mop with 0.73 confidence. This is likely because the fur of the hound is similar to the head of a mop. For the last example, PAR correctly predicts the object to be red wine with 0.59 confidence, but AlexNet predicts it to be a goblet with 0.74 confidence. This is likely because part of the image is indeed part of a goblet, but PAR may learn to make predictions based on the global concept considering the bottle, the liquid, and part of the goblet together.

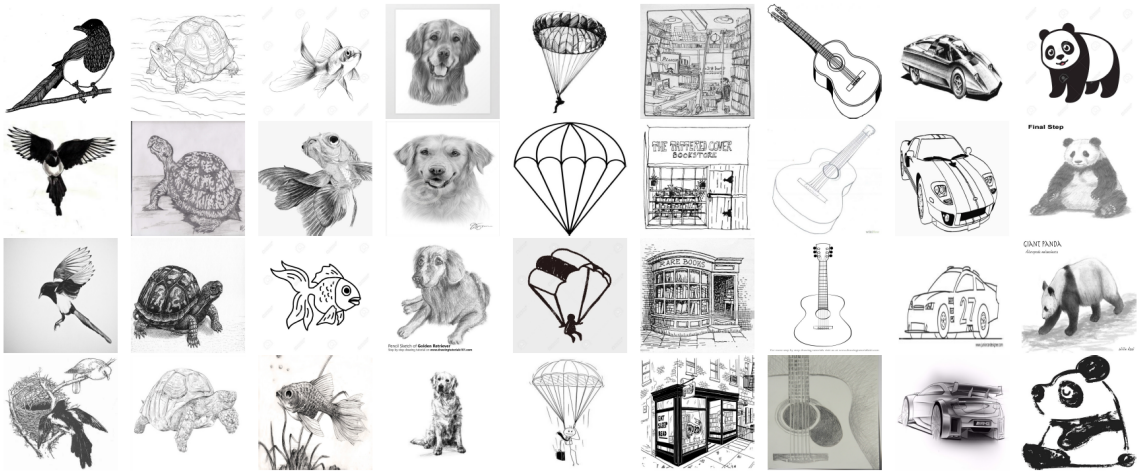


Figure 3.6: Sample Images from ImageNet-Sketch.

3.4 The ImageNet-Sketch Dataset

Inspired by the Sketch data of [91] with seven classes, and several other Sketch datasets, such as the *Sketchy* dataset [133] with 125 classes and the *Quick Draw!* dataset [123] with 345 classes, and motivated by absence of a large-scale sketch dataset fitting the shape and size of popular image classification benchmarks, we construct the ImageNet-Sketch data set for evaluating the out-of-domain classification performance of vision models trained on ImageNet.

Compatible with standard ImageNet validation data set for the classification task, our ImageNet-Sketch data set consists of 50000 images, 50 images for each of the 1000 ImageNet classes. We construct the data set with Google Image queries "sketch of ____", where ____ is the standard class name. We only search within the "black and white" color scheme. We initially query 100 images for every class, and then manually clean the pulled images by deleting the irrelevant images and images that are for similar but different classes. For some classes, there are less than 50 images after manually cleaning, and then we augment the data set by flipping and rotating the images.

We expect ImageNet-Sketch to serve as a unique ImageNet-scale out-of-domain evaluation dataset for image classification. Also, notably, different from perturbed ImageNet validation sets [41, 59], the images of ImageNet-Sketch are collected independently from the original validation images. The independent collection procedure is more similar to [126], who collected a new set of standard colorful ImageNet validation images. However, while their goal was to assess overfitting to the benchmark validation sets, and thus they tried replicate the ImageNet collection procedure exactly, our goald is to collect out-of-domain black-and-white sketch images with the goal of testing a model's ability to extrapolate out of domain.[5] Sample images are shown in Figure 3.6.

[5]The ImageNet-Sketch data can be found at: https://github.com/HaohanWang/ImageNet-Sketch

3.5 Conclusion

The chapter is a collection of methods developed to learn robust models by countering the superficial features. These empirical methods are validated by the empirical performances on the benchmark datasets of image classification, mostly in the domain generalization setup. These three methods fall into two different paradigms of solving this problem: to augment the data to dilute the superficial features and to regularize the hypothesis space so that the model will be incapable to learn a substantial amount of superficial features empirically. In addition, we notice that there is a common theme of these methods of countering the superficial features, and there may exist a principled view of these method development process. We will attempt to discuss this principled view in the next chapter.

Chapter 4

Principled Solutions of Learning Robust Models by Countering Superficial Features

This chapter is aimed to serve as a reflection of the development of the empirical methods. Since the development of the empirical methods all follow the the paradigm of asking the questions of what are the superficial features and how we can counter the models to learn them, we will try to formalize this paradigm in this chapter. Our formalization will lead to a new generalization bound, and the bound will naturally lead to a set of principled solutions, which are conveniently linked to the previously invented methods in this thesis. Finally, since all these methods will require an extra knowledge of what the superficial features are, as the last piece of this chapter, we will introduce a method that is absent of such knowledge, but can still do well empirically.

4.1 Background and Problem Setup

Our main goal is to quantify the generalization error bound of a model regarding the question that if we train a model with data from one distribution (*i.e.*, the source distribution), how can we guarantee the error to be small over other unseen, but related distributions (*i.e.*, target distributions). Quantifying the generalization error over two arbitrary distributions is not useful, thus, we require the distributions of study similar but different: being similar in the sense that there exists a common function that can achieve zero error over both distributions, while being different in the sense that there exists another different function that can only achieve zero error over the training distribution, but not the test distribution. We will formalize this "similar but different" property in the sequel.

There is a rich collection of theoretical discussions in quantifying the generalization error across distributions, mainly in the field of *unsupervised domain adaptation* [8, 9]. Most of these analyses, although in various forms [30, 42, 108, 185], mostly involve two additional terms in comparison to standard machine learning generalization bound: one term describes the "learnable" nature of the problem and one term quantifies the differences of the two distributions. This second term probably inspired most of the empirical methods forcing invariant representations from distributions, where "invariant" intuitively means the model's prediction preserves under certain shift of the data.

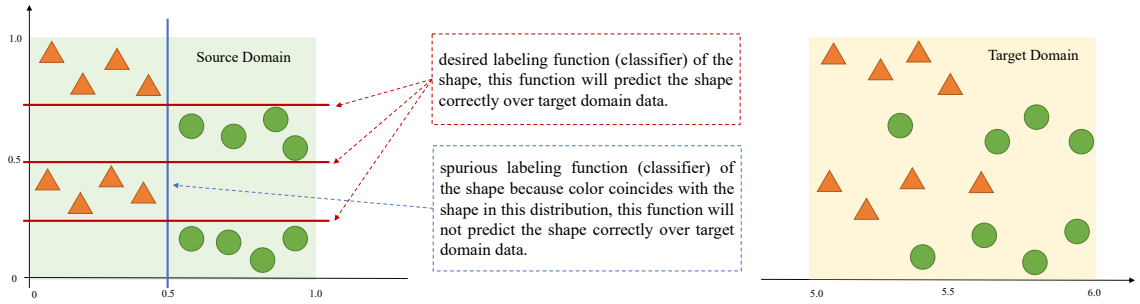


Figure 4.1: A toy example of the main problem focused in this section. as we aim to classify triangles vs. circles, the spurious correlation between color and shape in the training distribution will likely mislead the model to learn a spurious decision boundary (the distribution-specific labelling function), which may not be effective even if the marginals are aligned, but there exists another decision boundary (labeling function), which can classify the target distribution data correctly even if the marginals are not aligned.

Recently, the value of invariance is challenged [171, 187]. For example, Zhao et al. [187] argued that "invariance is not sufficient" by showing counter examples violating the "learnable" nature of the problem and formalized the understanding as that the two distributions have possibly different labeling functions.

However, we find the argument of disparity in labeling functions less intuitive, because human will nonetheless be able to agree on the label of an object whichever distribution the object lies in: in the context of this section, we argue a shared labeling function always exists (in any task reasonable to human), but the ERM model may not have the incentive to learn this function and learns a spurious one instead.

For example, our view of the challenges in this topic is illustrated with a toy example in Figure 4.1 where the model is trained on the source domain data to classify triangle vs. circle and tested on the target domain data. However, the color coincides with the shape on the source domain, so the model may learn either the desired function (relying on shape) or the spurious function (relying on color). The spurious function will not classify the target domain data correctly while the desired function can, but the ERM cannot differentiate them. As one may expect, whether shape or color is considered as desired or spurious is subjective dependent on the task or the data, and in general irrelevant to the statistical nature of the problem. Therefore, our error bound will require the knowledge of the spurious function. While this is a toy example, this scenario surly exists in real world tasks [*e.g.*, 41, 75, 166].

Therefore, we formalize the problem as learning against spurious functions, and argue that the central problem is still invariance, but instead of invariance to marginals, we urge for invariance to the spurious function.

Notations We consider a binary classification problem from feature space $\mathcal{X} \in \mathbb{R}^p$ to label space $\mathcal{Y} \in \{0, 1\}$. The distribution over $\mathcal{X}$ is denoted as $\mathbf{P}$. A *labeling function* $f : \mathcal{X} \to \mathcal{Y}$ is a function that maps feature $\mathbf{x}$ to its label $\mathbf{y}$. A *hypothesis* or *model* $\theta : \mathcal{X} \to \mathcal{Y}$ is also a function that maps feature to the label. The difference in naming is only because we want to differentiate

whether the function is a natural property of the space or distribution (thus called a labeling function) or a function to estimate (thus called a hypothesis or model). The hypothesis space is denoted as Θ. We use dom to denote the domain (input space) of a function, thus $\text{dom}(\theta) = \mathcal{X}$.

This work studies the generalization error across two distributions, namely source and target distribution, denoted as $\mathbf{P}_s$ and $\mathbf{P}_t$ respectively. We are only interested when these two distributions are, considered by a human, similar but different: being similar means there exists a *human-aligned labeling function*, f_h, that maps any $\mathbf{x} \in \mathcal{X}$ to its label (thus the label $\mathbf{y} := f_h(\mathbf{x})$); being different means there exists a *superficial labeling function*, f_m, that for any $\mathbf{x} \sim \mathbf{P}_s$, $f_m(\mathbf{x}) = f_h(\mathbf{x})$. This "similar but different" property will be reiterated as an assumption (**A2**) later. We use $(\mathbf{x}, \mathbf{y})$ to denote a sample, and use $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}}$ to denote a finite dataset if the features are drawn from $\mathbf{P}$. We use $\epsilon_{\mathbf{P}}(\theta)$ to denote the expected risk of θ over distribution $\mathbf{P}$, and use $\widehat{\cdot}$ to denote the estimation of the term $\cdot$ (*e.g.*, the empirical risk is $\widehat{\epsilon}_{\mathbf{P}}(\widehat{\theta})$). We use $l(\cdot, \cdot)$ to denote a generic loss function.

For a dataset $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}}$, if we train a model with

$$\widehat{\theta} = \arg\min_{\theta \in \Theta} \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}}} l(\theta(\mathbf{x}), \mathbf{y}), \tag{4.1}$$

previous generalization study suggests that we can expect the error rate to be bounded as

$$\epsilon_{\mathbf{P}}(\widehat{\theta}) \leqslant \widehat{\epsilon}_{\mathbf{P}}(\widehat{\theta}) + \phi(|\Theta|, n, \delta), \tag{4.2}$$

where $\epsilon_{\mathbf{P}}(\widehat{\theta})$ and $\widehat{\epsilon}_{\mathbf{P}}(\widehat{\theta})$ respectively are

$$\epsilon_{\mathbf{P}}(\widehat{\theta}) = \mathbb{E}_{\mathbf{x}\sim\mathbf{P}}|\widehat{\theta}(\mathbf{x}) - \mathbf{y}| = \mathbb{E}_{\mathbf{x}\sim\mathbf{P}}|\widehat{\theta}(\mathbf{x}) - f_h(\mathbf{x})| \quad \text{and} \quad \widehat{\epsilon}_{\mathbf{P}}(\widehat{\theta}) = \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}}} |\widehat{\theta}(\mathbf{x}) - \mathbf{y}|,$$

and $\phi(|\Theta|, n, \delta)$ is a function of hypothesis space $|\Theta|$, number of samples n, and δ accounts for the probability when the bound holds. This section only concerns with this generic form that can subsume many discussions, each with its own assumptions. We refer to these assumptions as **A1**.

A1: basic assumptions needed to derived (4.2), for example,

- when **A1** is "Θ is finite, $l(\cdot, \cdot)$ is a zero-one loss, samples are *i.i.d*", $\phi(|\Theta|, n, \delta) = \sqrt{(\log(|\Theta|) + \log(1/\delta))/2n}$
- when **A1** is "samples are *i.i.d*", $\phi(|\Theta|, n, \delta) = 2\mathcal{R}(\mathcal{L}) + \sqrt{(\log 1/\delta)/2n}$, where $\mathcal{R}(\mathcal{L})$ stands for Rademacher complexity and $\mathcal{L} = \{l_\theta \mid \theta \in \Theta\}$, where l_θ is the loss function corresponding to θ.

For more information or more concrete examples of the generic term, one can refer to relevant textbooks such as [13].

4.2 Generalization Bound of Learning Robust Models by Countering Superficial Features

Formally, we state the challenge of our human-aligned robust learning problem as the assumption:

A2: **Existence of Superficial Features:** For any $\mathbf{x} \in \mathcal{X}$, $\mathbf{y} := f_h(\mathbf{x})$. We also have a f_m that is different from f_h, and for $\mathbf{x} \sim \mathbf{P}_s$, $f_h(\mathbf{x}) = f_m(\mathbf{x})$.

Thus, the existence of f_m is a key reason to the lack of guarantees of the small empirical risk over $\mathbf{P}_s$ being generalized to $\mathbf{P}_t$, because θ that learns either f_h or f_m will lead to small source error, but only θ that learns f_h will lead to small target error. Note that f_m may not exist for an arbitrary $\mathbf{P}_s$. In other words, **A2** can be interpreted as a regulation to $\mathbf{P}_s$ so that f_m, while being different from f_h, exists for any $\mathbf{x} \sim \mathbf{P}_s$.

In this problem, f_m and f_h are not the same despite $f_m(\mathbf{x}) = f_h(\mathbf{x})$ for any $\mathbf{x} \sim \mathbf{P}_s$, and we consider the differences lie in the features they use. To describe this difference, we introduce the notation $\mathcal{A}(\cdot, \cdot)$, which denotes a set parametrized by the labeling function and the sample, to describe the *active set* of features that are used by the labeling function. By *active set*, we refer to the minimum set of features that a labeling function requires to map a sample to its label. Formally, we define

$$\mathcal{A}(f, \mathbf{x}) = \underset{\mathbf{z} \in \text{dom}(f), f(\mathbf{z}) = f(\mathbf{x})}{\arg\min} |\{i | \mathbf{z}_i = \mathbf{x}_i\}|, \tag{4.3}$$

where $|\cdot|$ measures the cardinality. Although $f_m(\mathbf{x}) = f_h(\mathbf{x})$, $\mathcal{A}(f_m, \mathbf{x})$ and $\mathcal{A}(f_h, \mathbf{x})$ can be different. $\mathcal{A}(f_m, \mathbf{x})$ is the *superficial features* following our definition.

Further, we define a new function difference given a sample as

$$d(\theta, f, \mathbf{x}) = \max_{\mathbf{z} \in \text{dom}(f): \mathbf{z}_{\mathcal{A}(f,\mathbf{x})} = \mathbf{x}_{\mathcal{A}(f,\mathbf{x})}} |\theta(\mathbf{z}) - f(\mathbf{z})|, \tag{4.4}$$

where $\mathbf{x}_{\mathcal{A}(f,\mathbf{x})}$ denotes the features of $\mathbf{x}$ indexed by $\mathcal{A}(f, \mathbf{x})$. In other words, the distance describes: given a sample $\mathbf{x}$, what is the maximum disagreement of the two functions θ and f for all the other data $\mathbf{z} \in \mathcal{X}$ with a constraint that the features indexed by $\mathcal{A}(f, \mathbf{x})$ are the same as those of $\mathbf{x}$. Notice that this difference is not symmetric, as the active set is determined by the second function. By definition, we have $d(\theta, f, \mathbf{x}) \geqslant |\theta(\mathbf{x}) - f(\mathbf{x})|$.

We introduce the following assumption:

A3: **Realized Hypothesis:** Given a large enough hypothesis space Θ, for any sample $(\mathbf{x}, \mathbf{y})$, for any $\theta \in \Theta$, which is not a constant mapping, if $\theta(\mathbf{x}) = \mathbf{y}$, then $d(\theta, f_h, \mathbf{x}) d(\theta, f_m, \mathbf{x}) = 0$

Intuitively, **A3** assumes θ at least learns one labeling function for the sample $\mathbf{x}$ if θ can map the $\mathbf{x}$ correctly.

Finally, to describe how θ depends on the active set of f, we introduce the term

$$r(\theta, \mathcal{A}(f, \mathbf{x})) = \max_{\mathbf{x}_{\mathcal{A}(f,\mathbf{x})} \in \text{dom}(f)_{\mathcal{A}(f,\mathbf{x})}} |\theta(\mathbf{x}) - \mathbf{y}|, \tag{4.5}$$

where $\mathbf{x}_{\mathcal{A}(f,\mathbf{x})} \in \text{dom}(f)_{\mathcal{A}(f,\mathbf{x})}$ denotes that the features of $\mathbf{x}$ indexed by $\mathcal{A}(f, \mathbf{x})$ are searched in the input space $\text{dom}(f)$. Notice that $r(\theta, \mathcal{A}(f, \mathbf{x})) = 1$ alone does not mean θ depends on the active set of f, it only means so when we also have $\theta(\mathbf{x}) = \mathbf{y}$ (see formal discussion in Lemma A2.1). With all above, we formalize a new generalization bound as follows:

Theorem 4.2.1. *With Assumptions **A1**-**A3**, $l(\cdot, \cdot)$ is a zero-one loss, with probability as least $1-\delta$, we have*

$$\epsilon_{\mathbf{P}_t}(\theta) \leqslant \widehat{\epsilon}_{\mathbf{P}_s}(\theta) + c(\theta) + \phi(|\Theta|, n, \delta) \tag{4.6}$$

where $c(\theta) = \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]r(\theta, \mathcal{A}(f_m, \mathbf{x}))$.

$\mathbb{I}[\cdot]$ is a function that returns 1 if the condition $\cdot$ holds true, and 0 otherwise. As θ may learn f_m, $\widehat{\epsilon}_{\mathbf{P}_s}(\theta)$ can no longer alone indicate $\epsilon_{\mathbf{P}_t}(\theta)$, thus we introduce $c(\theta)$ to account for the discrepancy. Intuitively, $c(\theta)$ quantifies the samples that are correctly predicted, but only because the θ learns f_m for that sample. $c(\theta)$ critically depends on the knowledge of f_m.

4.2.1 In Comparison to the View of Domain Adaptation

We further compare Theorem 4.2.1 with established understandings of domain adaptation. We summarize the several domain adaptation understandings [8, 9, 30, 42, 108, 185] in the following form:

$$\epsilon_{\mathbf{P}_t}(\theta) \leqslant \widehat{\epsilon}_{\mathbf{P}_s}(\theta) + D_\Theta(\mathbf{P}_s, \mathbf{P}_t) + \lambda + \phi'(|\Theta|, n, \delta) \tag{4.7}$$

where $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$ quantifies the differences of the two distributions, and λ describes the nature of the problem, and usually involves non-estimable terms about the problem or the distributions.

For example, [9] formalized the difference as $\mathcal{H}$-divergence, and described the corresponding empirical term as ($\Theta\Delta\Theta$ is the set of disagreement between two hypotheses in Θ):

$$D_\Theta(\mathbf{P}_s, \mathbf{P}_t) = 1 - \min_{\theta\in\Theta\Delta\Theta}\left(\frac{1}{n}\sum_{\mathbf{x}:\theta(\mathbf{x})=0} \mathbb{I}[\mathbf{x} \in (\mathbf{X}, \mathbf{Y})_{\mathbf{P}_s}] + \frac{1}{n}\sum_{\mathbf{x}:\theta(\mathbf{x})=1} \mathbb{I}[\mathbf{x} \in (\mathbf{X}, \mathbf{Y})_{\mathbf{P}_t}]\right), \tag{4.8}$$

where m denotes the number of unlabelled samples in $\mathbf{P}_s$ and $\mathbf{P}_t$ each. $\lambda = \epsilon_{\mathbf{P}_t}(\theta^\star) + \epsilon_{\mathbf{P}_s}(\theta^\star)$, where $\theta^\star = \arg\min_{\theta\in\Theta} \epsilon_{\mathbf{P}_t}(\theta) + \epsilon_{\mathbf{P}_s}(\theta)$,

In our formalization, as we assume the f_h applies to any $\mathbf{x} \in \mathcal{X}$ (according to **A2**), $\lambda = 0$ as long as the hypothesis space is large enough. Therefore, the difference mainly lies in the comparison between $c(\theta)$ and $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$. To compare them, we need an extra assumption:

A4: **Sufficiency of Training Samples:** for the two finite datasets in the study, *i.e.*, $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}_s}$ and $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}_t}$, for any $\mathbf{x} \in (\mathbf{X}, \mathbf{Y})_{\mathbf{P}_t}$, there exists one or many $\mathbf{z} \in (\mathbf{X}, \mathbf{Y})_{\mathbf{P}_s}$ such that

$$\mathbf{x} \in \{\mathbf{x}'|\mathbf{x}' \in \mathcal{X} \text{ and } \mathbf{x}'_{\mathcal{A}(f_h,\mathbf{z})} = \mathbf{z}_{\mathcal{A}(f_h,\mathbf{z})}\} \tag{4.9}$$

A4 intuitively means the finite training dataset needs to be diverse enough to describe the concept that needs to be learned. For example, imagine building a classifier to classify mammals *vs.* fishes from the distribution of photos to the distribution of sketches, we cannot expect the classifier to do anything good on dolphins if dolphins only appear in the test sketch dataset. **A4** intuitively regulates that if dolphins will appear in the test sketch dataset, they must also appear in the training dataset.

Now, in comparison to [9], we have

Theorem 4.2.2. *With Assumptions **A2**-**A4**, and if* $1 - f_h \in \Theta$*, we have*

$$c(\theta) \leqslant D_\Theta(\mathbf{P}_s, \mathbf{P}_t) + \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]r(\theta, \mathcal{A}(f_m, \mathbf{x})) \tag{4.10}$$

where $c(\theta) = \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]r(\theta, \mathcal{A}(f_m, \mathbf{x}))$ *and* $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$ *is defined as in* (4.8)*.*

The comparison involves an extra term, $q(\theta) := \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]r(\theta, \mathcal{A}(f_m, \mathbf{x}))$, which intuitively means that if θ learns f_m, how many samples θ can coincidentally predict correctly over the finite target set used to estimate $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$. For sanity check, if we replace $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}_t}$ with $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}_s}$, $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$ will be evaluated at 0 as it cannot differentiate two identical datasets, and $q(\theta)$ will be the same as $c(\theta)$. On the other hand, if no samples from $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}_t}$ can be mapped correctly with f_m (coincidentally), $q(\theta) = 0$ and $c(\theta)$ will be a lower bound of $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$.

The value of Theorem 4.2.2 lies in the fact that for an arbitrary target dataset $(\mathbf{X}, \mathbf{Y})_{\mathbf{P}_t}$, no samples out of which can be predicted correctly by learning f_m (a situation likely to occur for arbitrary datasets), $c(\theta)$ will always be a lower bound of $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$.

Further, when Assumption **A4** does not hold, we are unable to derive a clear relationship between $c(\theta)$ and $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$. The difference is mainly raised as a matter of fact that, intuitively, we are only interested in the problems that are "solvable" (**A4**, *i.e.*, hypothesis that used to reduce the test error in target distribution can be learned from the finite training samples) but "hard to solve" (**A2**, *i.e.*, another labeling function, namely f_m, exists for features sampled from the source distribution only), while $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$ estimates the divergence of two arbitrary distributions.

4.2.2 Estimation of $c(\theta)$

The estimation of $c(\theta)$ mainly involves two difficulties: the requirement of the knowledge of f_m and the computational cost of the search over the entire space $\mathcal{X}$.

The first difficulty is unavoidable by definition because the human-aligned learning has to be built upon the prior knowledge of what labeling function a human considers similar (what f_h is) or its opposite (what f_m is). Fortunately, in practice, we usually directly have the knowledge of $\mathcal{A}(f_m, \mathbf{x})$, *i.e.*, the superficial features, such as the texture of images.

The second difficulty is a computational problem, and we have a rich set of techniques to help circumvent it. For example, the search can be terminated once $r(\theta, \mathcal{A}(f_m, \mathbf{x}))$ is evaluated as 1 (*i.e.*, once we find a perturbation of superficial features that alters the prediction). This is similar to how adversarial attack [49] is used to evaluate the robustness of models. To further reduce the computational cost, one can consider to generate some out-of-domain data by perturbing superficial features beforehand, using these fixed perturbed data to evaluate may not be as accurate as using adversarial attack process, but it can usually reflect some flaws of the models that worth further attentions, as did by [41, 75, 166].

4.3 Principled Solutions

We continue to study how our formal analysis can lead to practical methods to learn human-aligned robust models. We first show that our formal result can naturally connect to existing methods for robust machine learning from both of those perspectives discussed. Further, as these methods all require some knowledge of assumptions of the superficial features, we continue to explore a new method that does not require so.

Theorem 4.2.1 suggests that to train a human-aligned robust model amounts to training for small $c(\theta)$ in addition to training for small empirical error (*i.e.*, $\widehat{\epsilon}_{\mathbf{P}_s}(\theta)$).

4.3.1 Connections to (Worst-case) Data Augmentation (Section 3.2)

We can consider the upper bound of $c(\theta)$

$$c(\theta) \leqslant \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} r(\theta, \mathcal{A}(f_m, \mathbf{x})) = \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \max_{\mathbf{x}_{\mathcal{A}(f_m,\mathbf{x})}\in \text{dom}(f_m)_{\mathcal{A}(f_m,\mathbf{x})}} |\theta(\mathbf{x}) - \mathbf{y}|, \quad (4.11)$$

which intuitively means that instead of $c(\theta)$ that studies only the correct predictions based on f_m, now we study any prediction based on f_m.

Further, as $|\theta(\mathbf{x}) - \mathbf{y}| \leqslant \max_{\mathbf{x}_{\mathcal{A}(f_m,\mathbf{x})}\in \text{dom}(f_m)_{\mathcal{A}(f_m,\mathbf{x})}} |\theta(\mathbf{x}) - \mathbf{y}|$, training for small (4.11) naturally subsumes training for small empirical loss. Therefore, we can directly train a model with

$$\min_{\theta} \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \max_{\mathbf{x}_{\mathcal{A}(f_m,\mathbf{x})}\in \text{dom}(f_m)_{\mathcal{A}(f_m,\mathbf{x})}} l(\theta(\mathbf{x}), \mathbf{y}), \quad (4.12)$$

which is to perturb the superficial features to maximize the training loss and then to solve the optimization problem with the perturbed samples. This is the worst-case data augmentation method [36] we discussed previously, and closely connected to the most widely accepted solution in the adversarial robust literature: the adversarial training [104]

While the above result shows that the method for learning human-aligned robust models is in mathematical connection to the method of worst-case data augmentation, in practice, a general application of this method will require some more assumptions.

In practice, when we use data augmentation to learn human-aligned models, we need either of the two following assumptions to hold:

A5-1: **Labeling Functions Separability of Features** For any $\mathbf{x} \in \mathcal{X}$, $\mathcal{A}(f_h, \mathbf{x}) \cap \mathcal{A}(f_m, \mathbf{x}) = \varnothing$

A5-2: **Labeling Functions Separability of Input Space** We redefine $f_m : \text{dom}(f_m) \to \mathcal{Y}$ and $\text{dom}(f_m) \subsetneq \mathcal{X}$. For any $\mathbf{x} \in \mathcal{X}$, $\max_{\mathbf{z}\in\text{dom}(f_m)\cap\text{dom}(f_h)} |f_h(\mathbf{z}) - f_h(\mathbf{z})| = 0$

While both of these assumptions appear strong, we believe a general discussion of human-aligned models may not be able to built without these assumptions. In particular, **A5-1** describes the situations that f'_h do not use the same set of features as f'_m. One example of this situation could be that the background of an image in dog vs. cat classification is considered features for f'_m, and the foreground of an image is considered as features for f'_h. **A5-2** describes the situations that while f'_m can uses the features that are considered by f'_h, the perturbation of the features within the domain of f'_m will not change the output of f'_h. One example of this situation could be that the texture of dog or cat in the dog vs. cat classification, while the texture can be perturbed, the perturbation cannot be allowed to an arbitrary scale of pixels (otherwise the perturbation is not a perturbation of texture). If neither of these assumptions holds, then the perturbation will be allowed to replace a dog's body with the one of a dolphin, and even human may not be able to confidently decide the resulting image is a dog, thus human-aligned learning will not be worth discussion.

4.3.2 Connections to Regularizing the Hypothesis Space (Section 3.3)

Connecting our theory to the other main thread is little bit tricky, as we need to extend the model to an encoder/decoder structure, where we use θ_e and θ_d to denote the encoder and decoder

respectively. Thus, by definition, we have $\theta(\mathbf{x}) = \theta_d(\theta_e(\mathbf{x}))$. Further, we define f'_m as the equivalent of f_m with the difference of operating on the representations $\theta_e(\mathbf{x})$. With the setup, optimizing the empirical loss and $c(\theta)$ leads to:

$$\min_{\theta_d,\theta_e} \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} l(\theta_d(\theta_e(\mathbf{x})),\mathbf{y}) - l(f'_m(\theta_e(\mathbf{x})),\mathbf{y}). \tag{4.13}$$

First, we need to formally introduce the properties regarding f'_m, as a correspondence to those of f_m.

Notations and Background with Encoder/Decoder Structure With the same binary classification problem from feature space $\mathcal{X}$ to label space $\mathcal{Y}$. We consider the encoder $\theta_e : \mathcal{X} \to \mathcal{E}$ and decoder $\theta_d : \mathcal{E} \to \mathcal{Y}$, $f' : \mathcal{E} \to \mathcal{Y}$ is the function that maps the embedding to the label.

Similarly, we introduce the assumptions on the $\mathcal{E}$ space.

A2': **Existence of Superficial Features:** For any $\mathbf{x} \in \mathcal{X}$ and an oracle encoder θ_e that $\mathbf{e} = \theta_e(\mathbf{x})$, $\mathbf{y} := f'_h(\mathbf{e})$. We also have a f'_m that is different from f'_h, and for $\mathbf{x} \sim \mathbf{P}_s$ and $\mathbf{e} = \theta_e(\mathbf{x})$, $f'_h(\mathbf{e}) = f'_m(\mathbf{e})$.

A3': **Realized Hypothesis:** Given a large enough hypothesis space Θ_d for decoders, for any sample $(\mathbf{x},\mathbf{y})$ and an encoder θ_e that $\mathbf{e} = \theta_e(\mathbf{x})$, for any $\theta_d \in \Theta_d$, which is not a constant mapping, if $\theta_d(\mathbf{e}) = \mathbf{y}$, then $d(\theta_d, f'_h, \mathbf{e})d(\theta_d, f'_m, \mathbf{e}) = 0$

With the above assumptions, following the same logic, we can derive the theorem corresponding to Theorem 3.1, with the only difference that how $c(\theta)$ is now derived.

Lemma 4.3.1. *With Assumptions **A1**, **A2'**, **A3'**, $l(\cdot,\cdot)$ is a zero-one loss, with probability as least $1-\delta$, we have*

$$\epsilon_{\mathbf{P}_t}(\theta) \leqslant \widehat{\epsilon}_{\mathbf{P}_s}(\theta) + c(\theta) + \phi(|\Theta|, n, \delta) \tag{4.14}$$

where $c(\theta) = \dfrac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] r(\theta_d, \mathcal{A}(f'_m, \theta_e(\mathbf{x})))$.

Now, we continue to show that how training for small $c(\theta)$ amounts to solving (4.13). To proceed, we need either of the two following assumptions to hold:

A5-1': **Labeling Functions Separability of Features** For any $\mathbf{x} \in \mathcal{X}$ and an encoder θ_e that $\mathbf{e} = \theta_e(\mathbf{x})$, $\mathcal{A}(f'_h, \mathbf{e}) \cap \mathcal{A}(f'_m, \mathbf{e}) = \varnothing$

A5-2': **Labeling Functions Separability of Input Space** We redefine $f'_m : \mathrm{dom}(f'_m) \to \mathcal{Y}$ and $\mathrm{dom}(f'_m) \subsetneq \mathcal{E}$. For any $\mathbf{x} \in \mathcal{X}$ and an encoder θ_e that $\mathbf{e} = \theta_e(\mathbf{x})$, $\max_{\mathbf{z}\in\mathrm{dom}(f'_m)\cap\mathrm{dom}(f'_h)} |f'_h(\mathbf{z}) - f'_h(\mathbf{z})| = 0$

Also, notice that, assumptions **A5-1'** and **A5-2'** also regulates the encoder to be reasonably good. In other words, these assumptions will not hold for arbitrary encoders.

Now, we continue to derive (4.13) from Lemma B.1 as the following:

$$
\begin{aligned}
c(\theta) &= \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \mathbb{I}[\theta_d(\theta_e(\mathbf{x})) = \mathbf{y}] r(\theta_d, \mathcal{A}(f'_m, \mathbf{x})) \\
&= \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \mathbb{I}[\theta_d(\theta_e(\mathbf{x})) = \mathbf{y}] \max_{\theta_e(\mathbf{x})_{\mathcal{A}(f'_m,\mathbf{x})} \in \mathrm{dom}(\theta_d)_{\mathcal{A}(f'_m,\mathbf{x})}} |\theta_d(\theta_e(\mathbf{x})) - \mathbf{y}| \\
&= \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \max_{\theta_e(\mathbf{x})_{\mathcal{A}(f'_m,\mathbf{x})} \in \mathrm{dom}(\theta_d)_{\mathcal{A}(f'_m,\mathbf{x})}} |f'_m(\theta_e(\mathbf{x})) - \mathbf{y}| \\
&\leqslant \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \max_{\theta_e(\mathbf{x})\in \mathrm{dom}(\theta_d)} |f'_m(\theta_e(\mathbf{x})) - \mathbf{y}|
\end{aligned}
$$

The third line is because of the definition of $\mathbb{I}[\theta_d(\theta_e(\mathbf{x})) = \mathbf{y}] r(\theta_d, \mathcal{A}(f'_m, \mathbf{x}))$ and assumptions of A3' and either **A5-1'** or **A5-2'**. Therefore, optimizing the empirical loss and $c(\theta)$ leads to

$$
\min_{\theta_d,\theta_e} \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} l(\theta_d(\theta_e(\mathbf{x})), \mathbf{y}) - l(f'_m(\theta_e(\mathbf{x})), \mathbf{y})
$$

Then the question left is how to get f'_m. We can design a specific architecture given the prior knowledge of the data, then f'_m can be directly estimated through $\min_{f'_m} \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} l(f'_m(\theta_e(\mathbf{x})), \mathbf{y})$, which connects to many methods in previous section, such as [162]. Alternatively, we can estimate f'_m guided by other labels (*e.g.*, domain ids, batch ids *etc.*), then we simply have

$$
\min_{f'_m} \frac{1}{n} \sum_{(\mathbf{x},\mathbf{t})\in(\mathbf{X},\mathbf{T})} l(f'_m(\theta_e(\mathbf{x})), \mathbf{t}),
$$

which connects to many of the methods in domain adaptation literature, especially [39].

4.4 The Self-Challenging Algorithm

While our analysis suggests that we cannot have a robust model without the knowledge of $\mathcal{S}$ or f_h, we continue to ask that what the best we can do without such knowledge. If we use $\mathcal{F}$ to denote the set $\{f_h, \mathcal{S}\}$ and use i to index its element, we can have the following upper bound to optimize

$$
c(\theta) \leqslant \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} r(\theta, \mathcal{A}(\mathcal{S}, \mathbf{x})) \leqslant \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \sum_i r(\theta, \mathcal{A}(\mathcal{F}_i, \mathbf{x})). \tag{4.15}
$$

By optimizing the RHS of (4.15), we aim to discourage the learning towards functions that *only* rely on one labelling function for each sample for *any* labelling functions. Intuitively, the method is to encourage the model's usage in *all* possible features (either associated with f_h or $\mathcal{S}$), thus the model may be more robust to the changes of features when dealing with perturbations of the data.

A model using all the features is not expected to be better than a method that only uses the f_h associated features. However, as many other methods require specific knowledge of the superficial features, we believe this method is a better practice than vanilla training when there are no side information, since we can use the model estimated in the previous iteration as a substitute of f_h or $\mathcal{S}$.

Therefore, ideally, our proposed algorithm can be sketched as the follows. At iteration t

- Identify $\mathcal{A}(\theta^{(t-1)}, \mathbf{x})$
- Sample $\mathbf{z} = \arg\max_{\mathbf{x}_{\mathcal{A}(\theta^{(t-1)},\mathbf{x})} \in \text{dom}(\theta^{(t-1)})_{\mathcal{A}(\theta^{(t-1)},\mathbf{x})}} |\theta^{(t-1)}(\mathbf{x}) - \mathbf{y}|$
- Continue to train the model with $\mathbf{z}$ to get $\theta^{(t)}$

where the second step is worst-case data augmentation.

Further, this idea can be straightforwardly extended to the encoder/decoder structure following the related discussion in the previous section. With θ_e and θ_d denoting the encoder and decoder respectively and $\mathbf{e} = \theta_e(\mathbf{x})$, we have the algorithm sketched for the encoder/decoder structure as

- Identify $\mathcal{A}(\theta_d^{(t-1)}, \mathbf{x})$
- Sample $\mathbf{z} = \arg\max_{\mathbf{e}_{\mathcal{A}(\theta_d^{(t-1)},\mathbf{e})} \in \text{dom}(\theta_d^{(t-1)})_{\mathcal{A}(\theta_d^{(t-1)},\mathbf{e})}} |\theta_d^{(t-1)}(\mathbf{e}) - \mathbf{y}|$
- Continue to train the model with $\mathbf{z}$ to get $\theta^{(t)}$

Inspired by these results, the next section introduces the self-challenging method that can empirically learn the robust models without any knowledge of the superficial features.

4.4.1 Background and Related Work

Imagine teaching a child to visually differentiate "dog" from "cat": when presented with a collection of illustrations from her picture books, she may immediately answer that "cats tend to have chubby faces" and end the learning. However, if we continue to ask for more differences, she may start to notice other features like ears or body-size. We conjecture this follow-up challenge question plays a significant role in helping human reach the remarkable generalization ability. Most people should be able to differentiate "cat" from "dog" visually even when the images are presented in irregular qualities. After all, we did not stop learning after we picked up the first clue when we were children, even the first clue was good enough to help us recognize all the images in our textbook.

Nowadays, deep neural networks have exhibited remarkable empirical results over various computer vision tasks, yet these impressive performances seem unmet when the models are tested with the samples in irregular qualities [166] (*i.e.*, out-of-domain data, samples collected from the distributions that are similar to, but different from the distributions of the training samples). To account for this discrepancy, technologies have been invented under the domain adaptation regime [9, 15], where the goal is to train a model invariant to the distributional differences between the source domain (*i.e.*, the distribution of the training samples) and the target domain (*i.e.*, the distribution of the testing samples) [22, 167].

As the influence of machine learning increases, the industry starts to demand the models that can be applied to the domains that are not seen during the training phase. Domain generalization

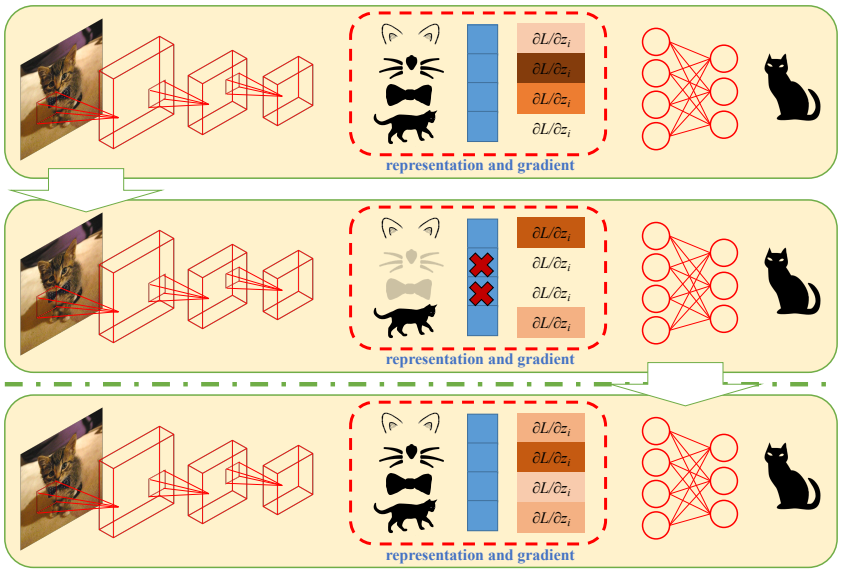


Figure 4.2: The essence of our Representation Self-Challenging (RSC) training method: top two panels: the algorithm mutes the feature representations associated with the highest gradient, such that the network is forced to predict the labels through other features; bottom panel: after training, the model is expected to leverage more features for prediction in comparison to models trained conventionally.

[114], as an extension of domain adaptation, has been studied as a response. The central goal is to train a model that can align the signals from multiple source domains.

Further, Wang *et al.* extend the problem to ask how to train a model that generalizes to an arbitrary domain with only the training samples, but not the corresponding domain information, as these domain information may not be available in the real world [163]. Our section builds upon this set-up and aims to offer a solution that allows the model to be robustly trained without domain information and to empirically perform well on unseen domains.

In this section, we introduce a simple training heuristic that improves cross-domain generalization. This approach discards the representations associated with the higher gradients at each epoch, and forces the model to predict with remaining information. Intuitively, in a image classification problem, our heuristic works like a "self-challenging" mechanism as it prevents the fully-connected layers to predict with the most predictive subsets of features, such as the most frequent color, edges, or shapes in the training data. We name our method Representation Self Challenging (RSC) and illustrate its main idea in Figure 4.2.

We present mathematical analysis that RSC induces a smaller generalization bound. We further demonstrate the empirical strength of our method with domain-agnostic cross-domain evaluations, following previous setup [163]. We also conduct ablation study to examine the alignment between its empirical performance and our intuitive understanding. The inspections also shed light upon the choices of its extra hyperparameter.

We summarize the related DG works from two perspectives: learning domain invariant features and augmenting source domain data. Further, as RSC can be broadly viewed as a generic training heuristic for CNN, we also briefly discuss the general-purpose regularizations that appear similar to our method.

DG through Learning Domain Invariant Features: These methods typically minimize

the discrepancy between source domains assuming that the resulting features will be domain-invariant and generalize well for unseen target distributions. Along this track, Muandet *et al*. employed Maximum Mean Discrepancy (MMD) [114]. Ghifary *et al*. proposed a multi-domain reconstruction auto-encoder [44]. Li *et al*. applied MMD constraints to an autoencoder via adversarial training [94].

Recently, meta-learning based techniques start to be used to solve DG problems. Li *et al*. alternates domain-specific feature extractors and classifiers across domains via episodic training, but without using inner gradient descent update [93]. Balaji *et al*. proposed MetaReg that learns a regularization function (e.g., weighted ℓ_1 loss) particularly for the network's classification layer, while excluding the feature extractor [6].

Further, recent DG works forgo the requirement of source domains partitions and directly learn the cross-domain generalizable representations through a mixed collection of training data. Wang *et al*. extracted robust feature representation by projecting out superficial patterns like color and texture [163]. Wang *et al*. penalized model's tendency in predicting with local features in order to extract robust globe representation [162]. RSC follows this more recent path and directly activates more features in all source domain data for DG without knowledge of the partition of source domains.

DG through Augmenting Source Domain: These methods augment the source domain to a wider span of the training data space, enlarging the possibility of covering the span of the data in the target domain. For example, An auxiliary domain classifier has been introduced to augment the data by perturbing input data based on the domain classification signal [136]. Volpi *et al*. developed an adversarial approach, in which samples are perturbed according to fictitious target distributions within a certain Wasserstein distance from the source [158]. A recent method with state-of-the art performance is JiGen [18], which leverages self-supervised signals by solving jigsaw puzzles.

Key difference: These approaches usually introduce a model-specific DG model and rely on prior knowledge of the target domain, for instance, the target spatial permutation is assumed by JiGen [18]. In contrast, RSC is a model-agnostic training algorithm that aims to improve the cross-domain robustness of any given model. More importantly, RSC does not utilize any knowledge of partitions of domains, either source domain or target domain, which is the general scenario in real world application.

Generic Model Regularization: CNNs are powerful models and tend to overfit on source domain datasets. From this perspective, model regularization, *e.g.*, weight decay [118], early stopping, and shake-shake regularization [40], could also improve the DG performance. Dropout [143] mutes features by randomly zeroing each hidden unit of the neural network during the training phase. In this way, the network benefit from the assembling effect of small subnetworks to achieve a good regularization effect. Cutout [29] and HaS [141] randomly drop patches of input images. SpatialDropout [149] randomly drops channels of a feature map. DropBlock [43] drops contiguous regions from feature maps instead of random units. DropPath [87] zeroes out an entire layer in training, not just a particular unit. MaxDrop [119] selectively drops features of high activations across the feature map or across the channels. Adversarial Dropout [120] dropouts for maximizing the divergence between the training supervision and the outputs from the network. [90] leverages Adversarial Dropout [120] to learn discriminative features by enforcing the cluster assumption.

4.4.2 Method

Notations: $(\mathbf{x}, \mathbf{y})$ denotes a sample-label pair from the data collection $(\mathbf{X}, \mathbf{Y})$ with n samples, and $\mathbf{z}$ (or $\mathbf{Z}$) denotes the feature representation of $(\mathbf{x}, \mathbf{y})$ learned by a neural network. $f(\cdot; \theta)$ denotes the CNN model, whose parameters are denoted as θ. $h(\cdot; \theta^{\text{top}})$ denotes the task component of $f(\cdot; \theta)$; $h(\cdot; \theta^{\text{top}})$ takes $\mathbf{z}$ as input and outputs the logits prior to a softmax function; θ^{top} denotes the parameters of $h(\cdot; \theta^{\text{top}})$. $l(\cdot, \cdot)$ denotes a generic loss function. RSC requires one extra scalar hyperparameter: the percentage of the representations to be discarded, denoted as p. Further, we use $\widehat{\cdot}$ to denote the estimated quantities, use $\tilde{\cdot}$ to denote the quantities after the representations are discarded, and use t in the subscript to index the iteration. For example, $\widehat{\theta}_t$ means the estimated parameter at iteration t.

Self-Challenging Algorithm

As a generic deep learning training method, RSC solves the same standard loss function as the ones used by many other neural networks, *i.e.*,

$$\widehat{\theta} = \arg\min_{\theta} \sum_{\langle \mathbf{x}, \mathbf{y} \rangle \sim \langle \mathbf{X}, \mathbf{Y} \rangle} l(f(\mathbf{x}; \theta), \mathbf{y}),$$

but RSC solves it in a different manner.

At each iteration, RSC inspects the gradient, identifies and then mutes the most predictive subset of the representation $\mathbf{z}$ (by setting the corresponding values to zero), and finally updates the entire model.

This simple heuristic has three steps (for simplicity, we drop the indices of samples and assume the batch size is 1 in the following equations):

1. Locate: RSC first calculates the gradient of upper layers with respect to the representation as follows:

$$\mathbf{g}_{\mathbf{z}} = \partial(h(\mathbf{z}; \widehat{\theta}_t^{\text{top}}) \odot \mathbf{y}) / \partial \mathbf{z}, \tag{4.16}$$

   where $\odot$ denotes an element-wise product. Then RSC computes the $(100 - p)^{\text{th}}$ percentile, denoted as q_p. Then it constructs a masking vector $\mathbf{m}$ in the same dimension of $\mathbf{g}$ as follows. For the i^{th} element:

$$\mathbf{m}(i) = \begin{cases} 0, & \text{if} \quad \mathbf{g}_{\mathbf{z}}(i) \geqslant q_p \\ 1, & \text{otherwise} \end{cases} \tag{4.17}$$

   In other words, RSC creates a masking vector $\mathbf{m}$, whose element is set to 0 if the corresponding element in $\mathbf{g}$ is one of the top p percentage elements in $\mathbf{g}$, and set to 1 otherwise.

2. Mute: For every representation $\mathbf{z}$, RSC masks out the bits associated with larger gradients by:

$$\tilde{\mathbf{z}} = \mathbf{z} \odot \mathbf{m} \tag{4.18}$$

Algorithm 1: RSC Update Algorithm

Input: data set $\langle \mathbf{X}, \mathbf{Y} \rangle$, percentage of representations to discard p, other configurations such as learning rate η, maximum number of epoches T, *etc*;

Output: Classifier $f(\cdot; \widehat{\theta})$;

random initialize the model $\widehat{\theta}_0$;

while $t \leqslant T$ **do**

  for *every sample (or batch)* $\mathbf{x}, \mathbf{y}$ **do**

    calculate $\mathbf{z}$ through forward pass;

    calculate $\mathbf{g}_{\mathbf{z}}$ with Equation 4.16;

    calculate q_p and $\mathbf{m}$ as in Equation 4.17;

    generate $\tilde{\mathbf{z}}$ with Equation 4.18;

    calculate gradient $\tilde{\mathbf{g}}_\theta$ with Equation 4.19 and Equation 4.20;

    update $\widehat{\theta}_{t+1}$ as a function of $\widehat{\theta}_t$ and $\tilde{\mathbf{g}}_\theta$

  end

end

3. Update: RSC computes the softmax with perturbed representation with

$$\tilde{\mathbf{s}} = \text{softmax}(h(\tilde{\mathbf{z}}; \widehat{\theta}_t^{\text{top}})), \tag{4.19}$$

and then use the gradient

$$\tilde{\mathbf{g}}_\theta = \partial l(\tilde{\mathbf{s}}, \mathbf{y}) / \partial \widehat{\theta}_t \tag{4.20}$$

to update the entire model for $\widehat{\theta}_{t+1}$ with optimizers such as SGD or ADAM.

We summarize the procedure of RSC in Algorithm 1. No that operations of RSC comprise of only few simple operations such as pooling, threshold and element-wise product. Besides the weights of the original network, no extra parameter needs to be learned.

Theoretical Evidence

To expand the theoretical discussion smoothly, we will refer to the "dog" vs. "cat" classification example repeatedly as we progress. The basic set-up, as we introduced in the beginning of this section, is the scenario of a child trying to learn the concepts of "dog" vs. "cat" from illustrations in her book: while the hypothesis "cats tend to have chubby faces" is good enough to classify all the animals in her picture book, other hypotheses mapping ears or body-size to labels are also predictive.

On the other hand, if she wants to differentiate all the "dogs" from "cats" in the real world, she will have to rely on a complicated combination of the features mentioned about. Our main motivation of this section is as follows: this complicated combination of these features is already illustrated in her picture book, but she does not have to learn the true concept to do well in her finite collection of animal pictures.

This disparity is officially known as "covariate shift" in domain adaptation literature: the conditional distribution (*i.e.*, the semantic of a cat) is the same across every domain, but the model may learn something else (*i.e.*, chubby faces) due to the variation of marginal distributions.

With this connection built, we now proceed to the theoretical discussion, where we will constantly refer back to this "dog" vs. "cat" example.

Background As the large scale deep learning models, such as AlexNet or ResNet, are notoriously hard to be analyzed statistically, we only consider a simplified problem to argue for the theoretical strength of our method: we only concern with the upper layer $h(\cdot; \theta^{\text{top}})$ and illustrate that our algorithm helps improve the generalization of $h(\cdot; \theta^{\text{top}})$ when $\mathbf{Z}$ is fixed. Therefore, we can directly treat $\mathbf{Z}$ as the data (features). Also, for convenience, we overload θ to denote θ^{top} within the theoretical evidence section.

We expand our notation set for the theoretical analysis. As we study the domain-agnostic cross-domain setting, we no longer work with *i.i.d* data. Therefore, we use $\mathcal{Z}$ and $\mathcal{Y}$ to denote the collection of distributions of features and labels respectively. Let Θ be a hypothesis class, where each hypothesis $\theta \in \Theta$ maps $\mathcal{Z}$ to $\mathcal{Y}$. We use a set $\mathcal{D}$ (or $\mathcal{S}$) to index $\mathcal{Z}$, $\mathcal{Y}$ and θ. Therefore, $\theta^\star(\mathcal{D})$ denotes the hypothesis with minimum error in the distributions specified with $\mathcal{D}$, but with no guarantees on the other distributions.

e.g., $\theta^\star(\mathcal{D})$ can be "cats have chubby faces" when $\mathcal{D}$ specifies the distribution to be picture book.

Further, $\theta^\star$ denotes the classifier with minimum error on every distribution considered. If the hypothesis space is large enough, $\theta^\star$ should perform no worse than $\theta^\star(\mathcal{D})$ on distributions specified by $\mathcal{D}$ for any $\mathcal{D}$.

e.g., $\theta^\star$ is the true concept of "cat", and it should predict no worse than "cats have chubby faces" even when the distribution is picture book.

We use $\widehat{\theta}$ to denote any ERM and use $\widehat{\theta}_{\text{RSC}}$ to denote the ERM estimated by the RSC method. Finally, following conventions, we consider $l(\cdot, \cdot)$ as the zero-one loss and use a shorthand notation $L(\theta; \mathcal{D}) = \mathbb{E}_{\langle \mathbf{z}, \mathbf{y} \rangle \sim \langle \mathcal{Z}(\mathcal{D}), \mathcal{Y}(\mathcal{D}) \rangle} l(h(\mathbf{z}; \theta), \mathbf{y})$ for convenience, and we only consider the finite hypothesis class case within the scope of this section, which leads to the first formal result:

Corollary 4.4.1. *If*

$$|e(\mathbf{z}(\mathcal{S}); \theta^\star_{\text{RSC}}) - e(\tilde{\mathbf{z}}(\mathcal{S}); \theta^\star_{\text{RSC}})| \leqslant \xi(p), \tag{4.21}$$

where $e(\cdot; \cdot)$ is a function defined as

$$e(\mathbf{z}; \theta^\star) := \mathbb{E}_{\langle \mathbf{z}, \mathbf{y} \rangle \sim \mathcal{S}} l(f(\mathbf{z}; \theta^\star); \mathbf{y})$$

and $\xi(p)$ is a small number and a function of RSC's hyperparameter p; $\tilde{\mathbf{z}}$ is the perturbed version of $\mathbf{z}$ generated by RSC, it is also a function of p, but we drop the notation for simplicity.

Before we continue the discussions, we first introduce some additional assumptions,

A1: Θ is finite; $l(\cdot, \cdot)$ is zero-one loss for binary classification.

The assumption leads to classical discussions on the *i.i.d* setting in multiple textbooks (*e.g.*, [110]). However, modern machine learning concerns more than the *i.i.d* setting, therefore, we need to quantify the variations between train and test distributions. Analysis of domain adaptation is discussed [9], but still relies on the explicit knowledge of the target distribution to quantify

the bound with an alignment of the distributions. The following discussion is devoted to the scenario when we do not have the target distribution to align.

Since we are interested in the $\theta^\star$ instead of the $\theta^\star(\mathcal{D})$, we first assume Θ is large enough and we can find a global optimum hypothesis that is applicable to any distribution, or in formal words:

A2: $L(\theta^\star; \mathcal{D}) = L(\theta^\star(\mathcal{D}); \mathcal{D})$ for any $\mathcal{D}$.

This assumption can be met when the conditional distribution $\mathbb{P}(\mathbf{Y}(\mathcal{D})|\mathcal{Z}(\mathcal{D}))$ is the same for any $\mathcal{D}$.

e.g., The true concept of "cat" is the same for any collection of images.

The challenge of cross-domain evaluation comes in when there exists multiple optimal hypothesis that are equivalently good for one distribution, but not every optimal hypothesis can be applied to other distributions.

e.g., For the distribution of picture book, "cats have chubby faces" can predict the true concept of "cat". A model only needs to learn one of these signals to reduce training error, although the other signal also exists in the data.

The follow-up discussion aims to show that RSC can force the model to learn multiple signals, so that it helps in cross-domain generalization.

Further, Assumption **A2** can be interpreted as there is at least some features $\mathbf{z}$ that appear in every distributions we consider. We use i to index this set of features. Assumption **A2** also suggests that $\mathbf{z}_i$ is *i.i.d.* (otherwise there will not exist $\theta^\star$) across all the distributions of interest (but $\mathbf{z}$ is not *i.i.d.* because $\mathbf{z}_{-i}$, where $-i$ denotes the indices other than i, can be sampled from arbitrary distributions).

e.g., $\mathbf{z}$ is the image; $\mathbf{z}_i$ is the ingredients of the true concept of a "cat", such as ears, paws, and furs; $\mathbf{z}_{-i}$ is other features such as "sitting by the window".

We use $\mathcal{O}$ to specify the distribution that has values on the i^{th}, but 0s elsewhere. We introduce the next assumption:

A3: Samples of any distribution of interest (denoted as $\mathcal{A}$) are perturbed version of samples from $\mathcal{O}$ by sampling arbitrary features for $\mathbf{z}_{-i}$: $\mathbb{E}_{\mathcal{A}}[\mathbb{E}_{\mathcal{S}}[\mathbf{z}]] = \mathbb{E}_{\mathcal{O}}[\mathbf{z}]$

Notice that this does not contradict with our cross-domain set-up: while Assumption **A3** implies that data from any distribution of interest is *i.i.d* (otherwise the operation $\mathbb{E}_{\mathcal{A}}[]$ is not valid), the cross-domain difficulty is raised when only different subsets of $\mathcal{A}$ are used for train and test. For example, considering $\mathcal{A}$ to be a uniform distribution of $[0, 1]$, while the train set is uniformly sampled from $[0, 0.5]$ and the test set is uniformly sampled from $(0.5, 1]$.

Corollary 4.4.2. *If Assumptions **A1**, **A2**, and **A3** hold, we have, with probability at least* $1 - \delta$

$$
\begin{aligned}
&L(\widehat{\theta}_{\text{RSC}}(\mathcal{S}); \mathcal{S}) - L(\theta^\star_{\text{RSC}}(\mathcal{S}); \mathcal{D}) \\
&\qquad \leqslant (2\xi(p) + 1)\sqrt{\frac{2(\log(2|\Theta_{\text{RSC}}|) + \log(2/\delta))}{n}}
\end{aligned}
$$

As the result shows, whether RSC will succeed depends on the magnitude of $\xi(p)$. The smaller $\xi(p)$ is, the tighter the bound is, the better the generalization bound is. Interestingly, if $\xi(p) = 0$, our result degenerates to the classical generalization bound of *i.i.d* data.

While it seems the success of our method will depend on the choice of Θ to meet Condition 4.21, we will show RSC is applicable in general by presenting it forces the empirical counterpart $\widehat{\xi}(p)$ to be small. $\widehat{\xi}(p)$ is defined as

$$\widehat{\xi}(p) := |h(\widehat{\theta}_{\text{RSC}}, \mathbf{z}) - h(\widehat{\theta}_{\text{RSC}}, \tilde{\mathbf{z}})|,$$

where the function $h(\cdot, \cdot)$ is defined as

$$h(\widehat{\theta}_{\text{RSC}}, \mathbf{z}) = \sum_{(\mathbf{z},\mathbf{y})\sim\mathcal{S}} l(f(\mathbf{z}; \widehat{\theta}_{\text{RSC}}); \mathbf{y}). \tag{4.22}$$

We will show $\widehat{\xi}(p)$ decreases at every iteration with more assumptions:

A4: Discarding the most predictive features will increase the loss at current iteration.

A5: The learning rate η is sufficiently small (η^2 or higher order terms are negligible).

Formally,

Corollary 4.4.3. *If Assumption **A4** holds, we can simply denote*

$$h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}_t) = \gamma_t(p) h(\widehat{\theta}_{\text{RSC}}(t), \mathbf{z}_t),$$

*where $h(\cdot, \cdot)$ is defined in Equation 4.22. $\gamma_t(p)$ is an arbitrary number greater than 1, also a function of RSC's hyperparameter p. Also, if Assumption **A5** holds, we have:*

$$\Gamma(\widehat{\theta}_{\text{RSC}}(t+1)) = \Gamma(\widehat{\theta}_{\text{RSC}}(t)) - (1 - \frac{1}{\gamma_t(p)})||\tilde{\mathbf{g}}||_2^2 \eta$$

where

$$\Gamma(\widehat{\theta}_{\text{RSC}}(t)) := |h(\widehat{\theta}_{\text{RSC}}(t), \mathbf{z}_t) - h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}_t)|$$

t denotes the iteration, $\mathbf{z}_t$ (or $\tilde{\mathbf{z}}_t$) denotes the features (or perturbed features) at iteration t, and $\tilde{\mathbf{g}} = \partial h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}_t) / \partial \widehat{\theta}_{\text{RSC}}(t)$

Notice that $\widehat{\xi}(p) = \Gamma(\widehat{\theta}_{\text{RSC}})$, where $\widehat{\theta}_{\text{RSC}}$ is $\widehat{\theta}_{\text{RSC}}(t)$ at the last iteration t. We can show that $\widehat{\xi}(p)$ is a small number because $\Gamma(\widehat{\theta}_{\text{RSC}}(t))$ gets smaller at every iteration. This discussion is also verified empirically, as shown in Figure 4.3.

The decreasing speed of $\Gamma(\widehat{\theta}_{\text{RSC}}(t))$ depends on the scalar $\gamma_t(p)$: the greater $\gamma_t(p)$ is, the faster $\Gamma(\widehat{\theta}_{\text{RSC}}(t))$ descends. Further, intuitively, the scale of $\gamma_t(p)$ is highly related to the mechanism of RSC and its hyperparameter p. For example, RSC discards the most predictive representations, which intuitively guarantees the increment of the empirical loss (Assumption **A4**).

Finally, the choice of p governs the increment of the empirical loss: if p is small, the perturbation will barely affect the model, thus the increment will be small; while if p is large, the perturbation can alter the model's response dramatically, leading to significant ascend of the loss. However, we cannot blindly choose the largest possible p because if p is too large, the model may not be able to learn anything predictive at each iteration.

In summary, we offer the intuitive guidance of the choice of hyperparamter p: for the same model and setting,

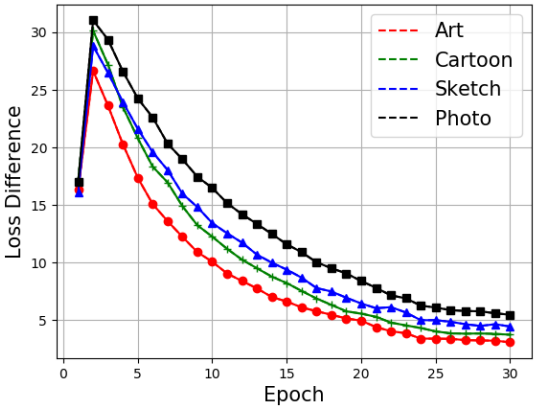


Figure 4.3: $\Gamma(\widehat{\theta}_{\text{RSC}}(t))$, i.e., "Loss Difference", plotted for the PACS experiment (details of the experiment setup will be discussed later). Except for the first epoch, $\Gamma(\widehat{\theta}_{\text{RSC}}(t))$ decreases consistently along the training process.

- the smaller p is, the smaller the training error will be;
- the bigger p is, the smaller the (cross-domain) generalization error (*i.e.*, difference between testing error and training error) will be.

Therefore, the success of our method depends on the choice of p as a balance of the above two goals.

Engineering Specification & Extensions

For simplicity, we detail the RSC implementation on a ResNet backbone + FC classification network. RSC is applied to the training phase, and operates on the last convolution feature tensor of ResNet. Denote the feature tensor of an input sample as $\mathbf{Z}$ and its gradient tensor of as $\mathbf{G}$. $\mathbf{G}$ is computed by back propagating the classification score with respect to the ground truth category. Both of them are of size $[7 \times 7 \times 512]$.

Spatial-wise RSC: In the training phase, global average pooling is applied along the channel dimension to the gradient tensor $\mathbf{G}$ to produce a weighting matrix w_i of size $[7 \times 7]$. Using this matrix, we select top p percentage of the $7 \times 7 = 49$ cells, and mute its corresponding features in $\mathbf{Z}$. Each of the 49 cells correspond to a $[1 \times 1 \times 512]$ feature vector in $\mathbf{Z}$. After that, the new feature tensor $\mathbf{Z}_{\mathbf{new}}$ is forwarded to the new network output. Finally, the network is updated through back-propagation. We refer this setup as spatial-wise RSC, which is the default RSC for the rest of this section.

Channel-wise RSC: RSC can also be implemented by dropping features of the channels with high-gradients. The rational behind the channel-wise RSC lies in the convolutional nature of DNNs. The feature tensor of size $[7 \times 7 \times 512]$ can be considered a decomposed version of input image, where instead of the RGB colors, there are 512 different characteristics of the each pixels. The C characteristics of each pixel contains different statistics of training data from that of the spatial feature statistics.

For channel-wise RSC, global average pooling is applied along the spatial dimension of $\mathbf{G}$, and produce a weighting vector of size $[1 \times 512]$. Using this vector, we select top p percentage of its 512 cells, and mute its corresponding features in $\mathbf{Z}$. Here, each of the 512 cells correspond to a $[7 \times 7]$ feature matrix in $\mathbf{Z}$. After that, the new feature tensor $\mathbf{Z}_{\mathbf{new}}$ is forwarded to the new network output. Finally, the network is updated through back-propagation.

Batch Percentage: Some dropout methods like curriculum dropout [111] do not apply dropout at the beginning of training, which improves CNNs by learning basic discriminative clues from unchanged feature maps. Inspired by these methods, we randomly apply RSC to some samples in each batch, leaving the other unchanged. This introduces one extra hyperparameter, namely Batch Percentage: the percentage of samples to apply RSC in each batch. We also apply RSC to top percentage of batch samples based on cross-entropy loss. This setup is slightly better than randomness.

Detailed ablation study on above extensions will be conducted in the experiment section below.

4.4.3 Experiments

Datasets

We consider the following four data collections as the battleground to evaluate RSC against previous methods.

- **PACS** [91]: seven classes over four domains (Artpaint, Cartoon, Sketches, and Photo). The experimental protocol is to train a model on three domains and test on the remaining domain.
- **VLCS** [150]: five classes over four domains. The domains are defined by four image origins, *i.e.*, images were taken from the PASCAL VOC 2007, LabelMe, Caltech and Sun datasets.
- **Office-Home** [154]: 65 object categories over 4 domains (Art, Clipart, Product, and Real-World).
- **ImageNet-Sketch** [162]: 1000 classes with two domains. The protocol is to train on standard ImageNet [129] training set and test on ImageNet-Sketch.

Ablation Study

We conducted five ablation studies on possible configurations for RSC on the PACS dataset [91]. All results were produced based on the ResNet18 baseline in [18] and were averaged over five runs.

(1) Feature Dropping Strategies (Table 4.1). We compared the two attention mechanisms to select the most discriminative spatial features. The "Top-Activiation" [119] selects the features with highest norms, whereas the "Top-Gradient" (default in RSC) selects the features with high gradients. The comparison shows that "Top-Gradient" is better than "Top-Activation", while both are better than the random strategy. Without specific note, we will use "Top-Gradient" as default in the following ablation study.

(2) Feature Dropping Percentage (choice of p) (Table 4.2): We ran RSC at different dropping percentages to mute spatial feature maps. The highest average accuracy was reached at $p = 33.3\%$. While the best choice of p is data-specific, our results align well with the theoretical discussion: the optimal p should be neither too large nor too small.

Feature Drop Strategies	backbone	artpaint	cartoon	sketch	photo	Avg ↑
Baseline [18]	ResNet18	78.96	73.93	70.59	**96.28**	79.94
Random	ResNet18	79.32	75.27	74.06	95.54	81.05
Top-Activation	ResNet18	80.31	76.05	76.13	**95.72**	82.03
Top-Gradient	ResNet18	**81.23**	**77.23**	**77.56**	95.61	**82.91**

Table 4.1: Ablation study of Spatial-wise RSC on Feature Dropping Strategies. Feature Dropping Percentage 50.0% and Batch Percentage 50.0%.

Feature Dropping Percentage	backbone	artpaint	cartoon	sketch	photo	Avg↑
66.7%	ResNet18	80.11	76.35	76.24	95.16	81.97
50.0%	ResNet18	81.23	77.23	77.56	95.61	82.91
33.3%	ResNet18	**82.87**	**78.23**	**78.89**	95.82	**83.95**
25.0%	ResNet18	81.63	78.06	78.12	96.06	83.46
20.0%	ResNet18	81.22	77.43	77.83	96.25	83.18
13.7%	ResNet18	80.71	77.18	77.12	**96.36**	82.84

Table 4.2: Ablation study of Spatial-wise RSC on Feature Dropping Percentage. We used "Top-Gradient" and fixed the Batch Percentage (50.0%) here.

Batch Percentage	backbone	artpaint	cartoon	sketch	photo	Avg↑
50.0%	ResNet18	**82.87**	78.23	78.89	95.82	83.95
33.3%	ResNet18	82.32	**78.75**	**79.56**	96.05	**84.17**
25.0%	ResNet18	81.85	78.32	78.75	**96.21**	83.78

Table 4.3: Ablation study of Spatial-wise RSC on Batch Percentage. We used "Top-Gradient" and fixed Feature Dropping Percentage (33.3%).

Method	backbone	artpaint	cartoon	sketch	photo	Avg↑
Spatial	ResNet18	82.32	78.75	79.56	**96.05**	84.17
Spatial+Channel	ResNet18	**83.43**	**80.31**	**80.85**	95.99	**85.15**

Table 4.4: Ablation study of Spatial-wise RSC verse Spatial+Channel RSC. We used the best strategy and parameter by Table 4.3:"Top-Gradient", Feature Dropping Percentage(33.3%) and Batch Percentage(33.3%).

Method	backbone	artpaint	cartoon	sketch	photo	Avg↑
Baseline [18]	ResNet18	78.96	73.93	70.59	**96.28**	79.94
Cutout[29]	ResNet18	79.63	75.35	71.56	95.87	80.60
DropBlock[43]	ResNet18	80.25	77.54	76.42	95.64	82.46
AdversarialDropout[120]	ResNet18	82.35	78.23	75.86	96.12	83.07
Random(S+C)	ResNet18	79.55	75.56	74.39	95.36	81.22
Top-Activation(S+C)	ResNet18	81.03	77.86	76.65	96.11	82.91
RSC: Top-Gradient(S+C)	ResNet18	**83.43**	**80.31**	**80.85**	95.99	**85.15**

Table 4.5: Ablation study of Dropout methods. "S" and "C" represent spatial-wise and channel-wise respectively. For fair comparison, results of above methods are report at their best setting and hyperparameters. RSC used the hyperparameters selected in above ablation studies:"Top-Gradient", Feature Dropping Percentage (33.3%) and Batch Percentage (33.3%).

(3) Batch Percentage (Table 4.3): RSC has the option to be only randomly applied to a subset of samples in each batch. Table 4.3 shows that the performance is relatively constant. Nevertheless we still choose 33.3% as the best option on the PACS dataset.

(4) Spatial-wise plus Channel-wise RSC (Table 4.4): In "Spatial+Channel", both spatial-wise and channel-wise RSC were applied on a sample at 50% probability, respectively. (Better options of these probabilities could be explored.) Its improvement over Spatial-wise RSC indicates that it further activated features beneficial to target domains.

PACS	backbone	artpaint	cartoon	sketch	photo	Avg ↑
Baseline[18]	AlexNet	66.68	69.41	60.02	89.98	71.52
Hex[163]	AlexNet	66.80	69.70	56.20	87.90	70.20
PAR[162]	AlexNet	66.30	66.30	64.10	89.60	72.08
MetaReg[6]	AlexNet	69.82	70.35	59.26	**91.07**	72.62
Epi-FCR[93]	AlexNet	64.70	72.30	65.00	86.10	72.00
JiGen[18]	AlexNet	67.63	71.71	65.18	89.00	73.38
MASF[32]	AlexNet	70.35	72.46	**67.33**	90.68	75.21
RSC(ours)	AlexNet	**71.62**	**75.11**	66.62	90.88	**76.05**
Baseline[18]	ResNet18	78.96	73.93	70.59	**96.28**	79.94
MASF[32]	ResNet18	80.29	77.17	71.69	94.99	81.03
Epi-FCR[93]	ResNet18	82.10	77.00	73.00	93.90	81.50
JiGen[18]	ResNet18	79.42	75.25	71.35	96.03	80.51
MetaReg[6]	ResNet18	**83.70**	77.20	70.30	95.50	81.70
RSC(ours)	ResNet18	83.43	**80.31**	**80.85**	95.99	**85.15**
Baseline[18]	ResNet50	86.20	78.70	70.63	97.66	83.29
MASF[32]	ResNet50	82.89	80.49	72.29	95.01	82.67
MetaReg[6]	ResNet50	87.20	79.20	70.30	97.60	83.60
RSC(ours)	ResNet50	**87.89**	**82.16**	**83.35**	**97.92**	**87.83**

Table 4.6: DG results on PACS[91] (Best in bold).

VLCS	backbone	Caltech	Labelme	Pascal	Sun	Avg ↑
Baseline[18]	AlexNet	96.25	59.72	70.58	64.51	72.76
Epi-FCR[93]	AlexNet	94.10	64.30	67.10	65.90	72.90
JiGen[18]	AlexNet	96.93	60.90	70.62	64.30	73.19
MASF[32]	AlexNet	94.78	**64.90**	69.14	67.64	74.11
RSC(ours)	AlexNet	**97.61**	61.86	**73.93**	**68.32**	**75.43**

Table 4.7: DG results on VLCS [150] (Best in bold).

(5) Comparison with different dropout methods (Table 4.5): Dropout has inspired a number of regularization methods for CNNs. The main differences between those methods lie in applying stochastic or non-stochastic dropout mechanism at input data, convolutional or fully connected layers. Results shows that our gradient-based RSC is better. We believe that gradient is an efficient and straightforward way to encode the sensitivity of output prediction. To the best of our knowledge, we compare with the most related works and illustrate the impact of gradients. (a) Cutout [29]. Cutout conducts random dropout on input images, which shows limited improvement over the baseline. (b) DropBlock [43]. DropBlock tends to dropout discriminative activated parts spatially. It is better than random dropout but inferior to non-stochastic dropout methods in Table 4.5 such as AdversarialDropout, Top-Activation and our RSC. (c) AdversarialDropout [90, 120]. AdversarialDropout is based on divergence maximization, while RSC is based on top gradients in generating dropout masks. Results show evidence that the RSC is more effective than AdversarialDropout. (d) Random and Top-Activation dropout strategies at their best hyperparameter settings.

Office-Home	backbone	Art	Clipart	Product	Real	Avg ↑
Baseline[18]	ResNet18	52.15	45.86	70.86	73.15	60.51
JiGen[18]	ResNet18	53.04	47.51	71.47	72.79	61.20
RSC(ours)	ResNet18	**58.42**	**47.90**	**71.63**	**74.54**	**63.12**

Table 4.8: DG results on Office-Home [154] (Best in bold).

ImageNet-Sketch	backbone	Top-1 Acc ↑	Top-5 Acc ↑
Baseline[163]	AlexNet	12.04	24.80
Hex[163]	AlexNet	14.69	28.98
PAR [162]	AlexNet	15.01	29.57
RSC(ours)	AlexNet	**16.12**	**30.78**

Table 4.9: DG results on ImageNet-Sketch [162].

Cross-Domain Evaluation

Through the following experiments, we used "Top-Gradient" as feature dropping strategy, 33.3% as Feature Dropping Percentages, 33.3% as Batch Percentage, and Spatial+Channel RSC. All

results were averaged over five runs. In our RSC implementation, we used the SGD solver, 30 epochs, and batch size 128. The learning rate starts with 0.004 for ResNet and 0.001 for AlexNet, learning rate decayed by 0.1 after 24 epochs. For PACS experiment, we used the same data augmentation protocol of randomly cropping the images to retain between 80% to 100%, randomly applied horizontal flipping and randomly (10% probability) convert the RGB image to greyscale, following [18].

In Tables 4.6,4.7,4.8, we compare RSC with the latest domain generalization work, such as Hex [163], PAR [162], JiGen [18] and MetaReg [6]. All these work only report results on different small networks and datasets. For fair comparison, we compared RSC to their reported performances with their most common choices of DNNs (*i.e.*, AlexNet, ResNet18, and ResNet50) and datasets. RSC consistently outperforms other competing methods.

The empirical performance gain of RSC can be better appreciated if we have a closer look at the PACS experiment in Table. 4.6. The improvement of RSC from the latest baselines [18] are significant and consistent: 4.5 on AlexNet, 5.2 on ResNet18, and 4.5 on ResNet50. It is noticeable that, with both ResNet18 and ResNet50, RSC boosts the performance significantly for sketch domain, which is the only colorless domain. The model may have to understand the semantics of the object to perform well on the sketch domain. On the other hand, RSC performs only marginally better than competing methods in photo domain, which is probably because that photo domain is the simplest one and every method has already achieved high accuracy on it.

4.4.4 Discussion

ImageNet	backbone	Top-1 Acc ↑	Top-5 Acc ↑	#Param. ↓
Baseline	ResNet50	76.13	92.86	25.6M
RSC(ours)	ResNet50	77.18	93.53	25.6M
Baseline	ResNet101	77.37	93.55	44.5M
RSC(ours)	ResNet101	78.23	94.16	44.5M
Baseline	ResNet152	78.31	94.05	60.2M
RSC(ours)	ResNet152	78.89	94.43	60.2M

Table 4.10: Generalization results on ImageNet. Baseline was produced with official Pytorch implementation and their ImageNet models.

Standard ImageNet Benchmark: With the impressive performance observed in the cross-domain evaluation, we further explore to evaluate the benefit of RSC with other benchmark data and higher network capacity.

We conducted image classification experiments on the Imagenet database[129]. We chose three backbones with the same architectural design while with clear hierarchies in model capacities: ResNet50, ResNet101, and ResNet152. All models were finetuned for 80 epochs with learning rate decayed by 0.1 every 20 epochs. The initial learning rate for ResNet was 0.01. All models follow extra the same training prototype in default Pytorch ImageNet implementation, using original batch size of 256, standard data augmentation and 224×224 as input size.

The results in Table 4.10 shows that RSC exhibits the ability reduce the performance gap between networks of same family but different sizes (*i.e.*, ResNet50 with RSC approaches the results of baseline ResNet101, and ResNet101 with RSC approaches the results of baseline ResNet151). The practical implication is that, RSC could induce faster performance saturation than increasing model sizes. Therefore one could scale down the size of networks to be deployed at comparable performance.

4.5 Conclusion

This chapter is a reflection of the development of the empirical methods, aiming to offer a summary of the development strategies of the empirical methods by countering superficial features, which can potentially inspire the development of the future development of the methods. Our formalization of the problem leads to a proved generalization bound on the problem of learning robust models by countering superficial features. The proved bound also leads to the discussion of a new method.

Chapter 5

Robustar: a Visual Interactive Toolbox to Counter Superficial Features with Human Supervision

Building upon the discussions of the previous chapters, we believe that one solution to learn the robust models by countering the superficial features lies in the key factor that one needs to be aware of the superficial features. Although we introduced an empirical method (Section 4.4) that can perform well on a collection of benchmark datasets for learning robust models, our theory (Section 4.2) suggests that a reliable, principled solution will inevitable require such knowledge.

This chapter aims to offer a solution to the practical challenge that we may not always have the knowledge of superficial features in image classification in practice to directly perform data augmentations or design regularizations. In this case, we can resort to the domain experts for the knowledge of superficial features. However, even when the domain experts are willing to offer the expertise, the knowledge may not easily be translated into an inductive bias or separable features. In addition, it may not be strategic to ask the domain expert use annotate the unuseful features for all the images in the training set.

Therefore, this chapter introduces an interactive toolbox that

- allows the domain experts to inspect how the model is making decisions and interact with the data by annotate the superficial features exploited by the model.
- guide the domain experts' focus to pay particular attention to the samples and features that leads to the model's learning of superficial features, leaving other samples untouched so that annotators' attention is needed at a minimum scale.
- offers convenient interface to allow the users directly continue to train or finetune the model with annotated data.

5.1 Overview

Figure 5.1 shows the overview process of the software. The software can work with models trained elsewhere, as long as these are standard vision models and have the standard pytorch

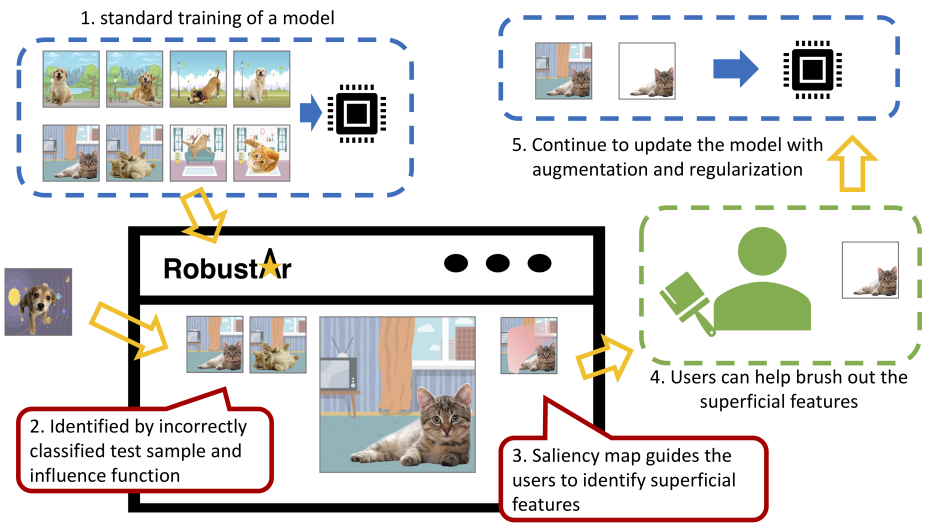


Figure 5.1: The working flow of the Robustar software: 1. the model can be trained elsewhere and then fed into the software; 2. With new test samples, the model can help identify the samples that are responsible for the prediction through influence function; 3. the software offers saliency map to help the user know which part of the features the model are paying necessary attention; 4. the users can use the drawing tools to brush out the superficial pixels; 5. new annotation of these images will serve as the role as augmented images for continued training.

checkpoints available. One can also edit our source code to integrate other vision models. If the influence function results are also calculated, the software can guide the users to the samples that are believed to be responsible for the incorrect classification of the images. Further, our software will also guide the user's attention of the features by the saliency-style interpretation of the models. Then the users can user canvas tool to annotate the superficial pixels of the images. Finally, with the newly annotated superficial features, we can continue to update the model with data augmentation and consistency regularization to help us discard these superficial features. The detailed training strategy has been discussed in Section 3.2.

5.2 Setup

As the goal of this system is to calibrate the model's prediction indeed aligns to the goal of the study, instead of exploiting some superficial features that correlate with the labels within a given dataset, we need to start with a model with reasonably small training and validation errors. Users can either upload such a model to the system or use the functions offered by the system to train a new model. In either case, the whole training dataset is expected upload also for the calibration. Also, we recommend the users to consider a model boosted by the adversarial training [104] technique, which is one of the most effective methods that can defend the model's against minor texture shifts.

To take the most advantage of the system, we recommend the user to use out-of-domain test

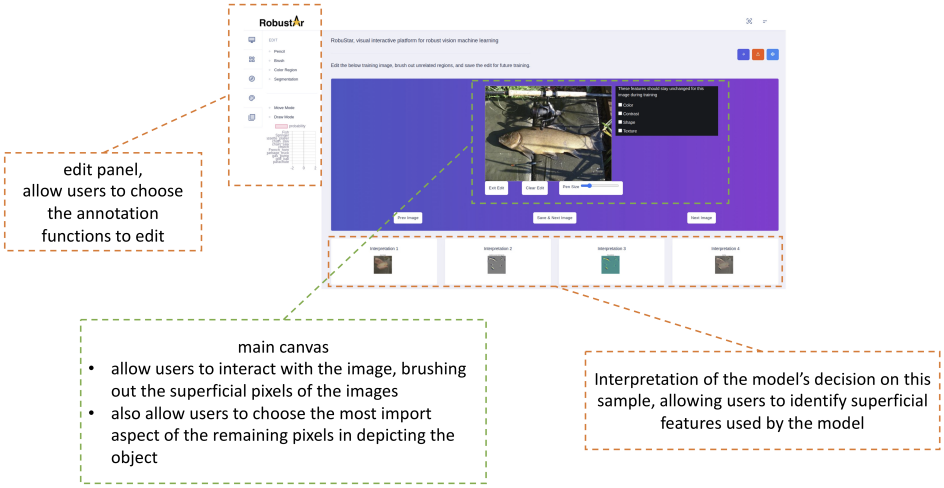


Figure 5.2: The main panel and its major functions for users to annotate the superficial features.

samples (*i.e.*, test samples from an independent data collection instead of from a cross-validated split), because the test data from the same collection (distribution) with training data tend to share the same superficial features with training data, thus may be inadequate to help identify the superficial features. As long as the test samples can represent a wider range of distributions, we do not need a large number of test samples

5.3 Identifying Superficial Features

Identifying Misleading Samples Ideally, to achieve a robust learning system, the signal from every sample needs to be examined and calibrated by human, which requires a potentially unrealistic working load. Thus, we have to rely on the system to propose the suspiciously misleading samples first.

The central assumption that enables the automatic proposal is that the superficial features are not shared between training samples and testing samples, so that when a test sample is misclassified, it is mostly due to the fact that the model learns a biased signal that accounts for the misclassification. This allows us to identify the samples with superficial features (*i.e.*, the samples that contribute to the misclassification.) Influence function [80] conveniently allows us to identify the samples that account for the misclassification.

Interpret Model's Decision With the identification of misleading samples, we continue to understand which part of the image sample deceives the model's decision. This process requires the neural network interpretation methods. We notice that, if the model is invariant to simple texture variations (*e.g.*, boosted by adversarial training), the pioneering model-interpretation methods, activation maximization [35] can sufficiently fulfill our goal.

Identify Superficial Features with Human Supervision The above analysis interprets which signals the model is leveraging to make a prediction. The human annotator needs to check further about whether the model predicts based on superficial features, and if so, the annotator needs to help pinpoint the superficial features in the data and so that our system can help to erase it.

If any of the interpretation area overlaps with the bias signal of the data, the annotation needs to use our system to select that area through our system. The interface offers the function for the annotator to "brush" out the pixels of the superficial features. To further help the annotator in the process of selecting the area of superficial features, we first use a semantic segmentation model to offer proposals of areas. If a segmented areas consists nothing but superficial features, the annotators can directly select that area, instead of selecting each pixels.

Randomized the identified signals After the annotators help identify the biased signal of the data, we need to further train our system to ignore those systems. The most straightforward step is to augment the data with the superficial features replaced with random patterns. To create a random pattern, we simply use randomize the pixel values of any identified areas.

5.4 Model Update

Now with the newly augmented data, we can continue to finetune the model with new augmented data and consistency regularization, following the techniques discussed in Section 3.2 The software also offers a panel for the users to update the hyperparamters used for training.

5.5 Conclusion

In the case when one does not know the superficial features explicitly in image classification, we offer a software for the users to inspect the training data and annotate the superficial image features. Further, our software can also help to identify the samples that may mislead the model and can help pinpoint the features that are used by the model to reduce the workload of the users. Our software can be found as https://github.com/HaohanWang/Robustar. With Docker installed, our software can be installed and run with two commands.

Chapter 6

Conclusion

In this thesis, we studied the problem of learning robust models with a narrow focus on the hypothesis that a reason of the non-robust behavior of learning robust models lies in the fact that there are superficial features that is undesired for the model to learn. We study the problem with different technical perspectives.

We first validated the hypothesis listed in Figure 1.1 and discussed some empirical observations. We also leveraged the empirical observation to explain several interesting machine learning behaviors.

Then, over the battleground of image classification, we discussed multiple methods to compete with previous SOTA methods. The main evaluation metric is cross-domain test accuracy, with a scenario we propose and refer to as *domain generalization without domain IDs*. We also introduced the ImageNet-Sketch dataset to test the robustness of an image classification model.

We continued to look at the theoretical foundation of this problem, proving a new generalization bound when the model is trained over data with superficial features. The formalization helps explain the performance drop of when the models are tested with other dataset. Also, the formalization are interestingly connected to the methods we developed in the previous section.

Finally, we contributed a toolbox with graphic user interface that allows users to examine how an image classification model perceives the data and to annotate the superficial features the model exploits.

Appendix A

Appendix

A1 Proofs for Chapter 3.2: Data Augmentation & Consistency Loss

A1.1 Assumptions and Validations

Assumptions on Data Augmentation Functions

We first regulate some basic properties of the data transformation functions used. Intuitively, we will consider the following three properties.

A1: "Dependence-preservation" with two perspectives: Label-wise, the transformation cannot alter the label of the data, which is a central requirement of almost all the data augmentation functions in practice. Feature-wise, the transformation will not introduce new dependencies between the samples.

A2: "Efficiency": the augmentation should only generate new samples of the same label as minor perturbations of the original one. If a transformation violates this property, there should exist other simpler transformations that can generate the same target sample.

A3: "Vertices": There are extreme cases of the transformations. For example, if one needs the model to be invariant to rotations from 0° to 60°, we consider the vertices to be 0° rotation function (thus identity map) and 60° rotation function. In practice, one usually selects the transformation vertices with intuitions.

These properties are formally introduced as assumptions below.

A1: Dependence-preservation: the transformation function will not alter the dependency regarding the label (*i.e.*, for any $a() \in \mathcal{A}$, $a(\mathbf{x})$ will have the same label as $\mathbf{x}$) or the features (*i.e.*, $a_1(\mathbf{x}_1)$ and $a_2(\mathbf{x}_2)$ are independent if $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent).

Remarks of **A1**:

- We consider the label-wise half of this argument as a fundamental property of any data augmentations. It has to be always true for data augmentation to be a useful technique.
- The feature-wise half of this argument is a fundamental property required to derive the generalization error bounds. Intuitively, it should hold for most data augmentation techniques in practice.

A2: Efficiency: for $\widehat{\theta}$ and any $a() \in \mathcal{A}$, $f(a(\mathbf{x}); \widehat{\theta})$ is closer to $\mathbf{x}$ than any other samples under a distance metric $d_e(\cdot, \cdot)$, *i.e.*, $d_e(f(a(\mathbf{x}); \widehat{\theta}), f(\mathbf{x}; \widehat{\theta})) \leqslant \min_{\mathbf{x}' \in \mathbf{X}_{-\mathbf{x}}} d_e(f(a(\mathbf{x}); \widehat{\theta}), f(\mathbf{x}'; \widehat{\theta}))$. We define $d_e(\cdot, \cdot)$ to be ℓ_1 norm.

Validation of **A2**:

- We test the assumption with MNIST data and rotation experiment. A2 essentially states the distance $d_e(\cdot, \cdot)$ is the smaller between a sample and its augmented copy (60° rotation) than the sample and the augmented copy from any other samples. We take 1000 training examples and calculate the ℓ_1 pair-wise distances between the samples and its augmented copies, then we calculated the frequencies when the A2 hold for one example. We repeat this for three different models, the vanilla model, the model trained with augmented data, and the model trained with regularized adversarial training. The results are shown in the

Table A1 and suggest that, although the A2 does not hold in general, it holds for regularized adversarial training case, where A2 is used. Further, we test the assumption in a more challenging case, where half of the training samples are 15° rotations of the other half, thus we may expect the A2 violated for every sample. Finally, as A2 is essentially introduced to replace the empirical Wasserstein distance with ℓ_1 distances of the samples and the augmented copies, we directly compare these metrics. However, as the empirical Wasserstein distance is forbiddingly hard to calculate (as it involves permutation statistics), we use a greedy heuristic to calculate by iteratively picking the nearest neighbor of a sample and then remove the neighbor from the pool for the next sample. Our inspection suggests that, even in the challenging scenario, the paired distance is a reasonably good representative of Wasserstein distance for regularized adversarial training method.

	Standard Scenario			Challenging Scenario		
	Vanilla	Augmented	Regularized	Vanilla	Augmented	Regularized
Frequency when A2 holds	0.005	0.152	0.999	0.001	0.021	0.711
Paired Distance	217968.06	42236.75	1084.4	66058.4	28122.45	4287.31
Wasserstein (greedy)	152736.47	38117.77	1084.4	37156.5	20886.7	4218.53
Paired/Wasserstein	1.42	1.10	1	1.77	1.34	1.02

Table A1: Empirical results from synthetic data for Assumption A2.

A3: Vertices: For a model $\widehat{\theta}$ and a transformation $a()$, we use $\mathbf{P}_{a,\widehat{\theta}}$ to denote the distribution of $f(a(\mathbf{x});\widehat{\theta})$ for $(\mathbf{x},\mathbf{y}) \sim \mathbf{P}$. "Vertices" argues that exists two extreme elements in $\mathcal{A}$, namely a^+ and a^-, with certain metric $d_x(\cdot,\cdot)$, we have

$$d_x(\mathbf{P}_{a^+,\widehat{\theta}}, \mathbf{P}_{a^-,\widehat{\theta}}) = \sup_{a_1,a_2\in\mathcal{A}} d_x(\mathbf{P}_{a_1,\widehat{\theta}}, \mathbf{P}_{a_2,\widehat{\theta}})$$

We define $d_x(\cdot,\cdot)$ to be Wasserstein-1 metric.

Discussion of **A3**:

- Notice that we do not need to argue that **A3** always holds. All we need is that **A3** can sometimes hold, and when it holds, we can directly train with the regularized vertex augmentation. Thus, anytime RVA empirically performs well is a favorable argument for A3. To show that RVA can sometimes perform well, we compare the RVA with vanilla (non-regularized) worst-case data augmentation (VWA) method across our synthetic experiment setup. We notice that out of six total scenarios ($\{$texture, rotation, contrast$\}$ $\times$ $\{$MNIST, CIFAR10$\}$), RVA outperforms VWA frequently (Table A2). This suggests that the domain-knowledge of vertices can actually help in most cases, although not guaranteed in every case.

Assumptions on Background and Generalization Error Bound

We first summarize a thread of previous analyses for error bounds in an extremely abstract manner. When the test data and train data are from the same distribution, many previous analyses can be sketched as:

$$r_{\mathbf{P}}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + \phi(|\Theta|, n, \delta) \tag{A.1}$$

Table A2: Results to show that **A3** can sometimes hold.

		Texture			Rotation			Invariance		
		Acc.	Rob.	Inv.	Acc.	Rob.	Inv.	Acc.	Rob.	Inv.
MNIST	RVA	99.1	99	100	99.3	95.1	65.4	99.4	97.4	41.3
	VWA	99.2	99.3	99.4	94.4	94.1	62.8	98.2	98.4	40.2
CIFAR10	RVA	63.5	62.2	100	75.9	47.7	67.8	76.7	69.3	57.4
	VWA	60.5	58.1	100	71.4	61.7	91.7	74.8	63.2	34.3

which states that the expected risk $r_{\mathbf{P}}(\widehat{\theta})$ can be bounded by the empirical risk $\widehat{r}_{\mathbf{P}}(\widehat{\theta})$ and a function of hypothesis space $|\Theta|$ and number of samples n; δ accounts for the probability when the bound holds. $\phi()$ is a function of these three terms. Dependent on the details of different analyses, different concrete examples of this generic term will need different assumptions. We use a generic assumption **A4** to denote the assumptions required for each example (Appendix A1.1).

Following our main goal to study how consistency loss and data augmentation help in accuracy, robustness, and invariance, our strategy in theoretical analysis is to derive error bounds for accuracy and robustness, and the error bound directly contains terms to regularize the invariance. Further, as robustness naturally bounds accuracy (*i.e.*, $r_{\mathbf{P}}(\widehat{\theta}) \leqslant r_{\mathbf{P},\mathcal{A}}(\widehat{\theta})$ following the definitions in (3.1) and (3.2) respectively), we only need to study the robust error.

To study the robust error, we need two additional technical assumptions. **A5** connects the worst distribution of expected risk and the worst distribution of the empirical risk, and **A6** connects the 0-1 classification error and cross-entropy error. Details of these assumptions are in Appendix A1.1.

A4: We list two classical examples here:

- when **A4** is "Θ is finite, $l(\cdot,\cdot)$ is a zero-one loss, samples are *i.i.d*", $\phi(|\Theta|, n, \delta) = \sqrt{(\log(|\Theta|) + \log(1/\delta))/2n}$
- when **A4** is "samples are *i.i.d*", $\phi(|\Theta|, n, \delta) = 2\mathcal{R}(\mathcal{L}) + \sqrt{(\log 1/\delta)/2n}$, where $\mathcal{R}(\mathcal{L})$ stands for Rademacher complexity and $\mathcal{L} = \{l_\theta \,|\, \theta \in \Theta\}$, where l_θ is the loss function corresponding to θ.

For more information or more concrete examples of the generic term, one can refer to relevant textbooks such as [13].

Remarks of **A4**:

- A4 stands for the fundamental assumptions used to derive standard generalization bounds. We rely on this assumption as how previous analytical works rely on it.

A5: the worst distribution for expected risk equals the worst distribution for empirical risk, *i.e.*,

$$\underset{\mathcal{P}' \in T(\mathbf{P},\mathcal{A})}{\arg\max}\, r_{\mathbf{P}'}(\widehat{\theta}) = \underset{\mathcal{P}' \in T(\mathbf{P},\mathcal{A})}{\arg\max}\, \widehat{r}_{\mathbf{P}'}(\widehat{\theta})$$

where $T(\mathbf{P}, \mathcal{A})$ is the collection of distributions created by elements in $\mathcal{A}$ over samples from $\mathbf{P}$.

Motivation of **A5**:

- Eq. (3.2) (in main paper) can be written equivalently into the expected risk over a pseudo distribution $\mathbf{P}'$ (see Lemma 1 in [152]), which is the distribution that can sample the data leading to the worst expected risk. Thus, equivalently, we can consider $\sup_{\mathcal{P}' \in T(\mathbf{P},\mathcal{A})} r_{\mathbf{P}'}(\widehat{\theta})$ as a surrogate of $r_{\mathbf{P},\mathcal{A}}(\widehat{\theta})$, where $T(\mathbf{P}, \mathcal{A})$ denotes the set of possible resulting distributions. Following the empirical strength of techniques such as adversarial training [104], we introduce an assumption relating the worst distribution of expected risk and the worst distribution of the empirical risk (namely, **A5**, in Appendix A1.1). Thus, the bound of our interest (*i.e.*, $\sup_{\mathbf{P}' \in T(\mathbf{P},\mathcal{A})} r_{\mathbf{P}'}(\widehat{\theta})$) can be analogously analyzed through $\sup_{\mathbf{P}' \in T(\mathbf{P},\mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta})$.

Discussion of **A5**:

- Assumption **A5** appears very strong, however, the successes of methods like adversarial training [104] suggest that, in practice, **A5** might be much weaker than it appears.

A6: With $(\mathbf{x}, \mathbf{y}) \in (\mathbf{X}, \mathbf{Y})$, the worst case sample in terms of maximizing cross-entropy loss and worst case sample in terms of maximizing classification error for model $\widehat{\theta}$ follows:

$$\forall \mathbf{x}, \quad \frac{\mathbf{y}^\top f(\mathbf{x}; \widehat{\theta})}{\inf_{a \in \mathcal{A}} \mathbf{y}^\top f(a(\mathbf{x}); \widehat{\theta})} \geqslant \exp\left(\mathbb{I}(g(f(\mathbf{x}; \widehat{\theta})) \neq g(f(\mathbf{x}'; \widehat{\theta})))\right) \tag{A.2}$$

where $\mathbf{x}'$ stands for the worst case sample in terms of maximizing classification error, *i.e.*,

$$\mathbf{x}' = \arg\min_{\mathbf{x}} \mathbf{y}^\top g(f(\mathbf{x}; \widehat{\theta}))$$

Also,

$$\forall \mathbf{x}, \quad |\inf_{a \in \mathcal{A}} \mathbf{y}^\top f(a(\mathbf{x}); \widehat{\theta})| \geqslant 1 \tag{A.3}$$

Intuitive understanding of **A6**:

- Although Assumption **A6** appears complicated, it describes simple situations that we will unveil in two scenarios:
  - If $g(f(\mathbf{x}; \widehat{\theta})) = g(f(\mathbf{x}'; \widehat{\theta}))$, which means either the sample is misclassified by $\widehat{\theta}$ or the adversary is incompetent to find a worst case transformation that alters the prediction, the RHS of Eq. A.2 is 1, thus Eq. A.2 always holds (because $\mathcal{A}$ has the identity map as one of its elements).
  - If $g(f(\mathbf{x}; \widehat{\theta})) \neq g(f(\mathbf{x}'; \widehat{\theta}))$, which means the adversary finds a transformation that alters the prediction. In this case, A2 intuitively states that the $\mathcal{A}$ is reasonably rich and the adversary is reasonably powerful to create a gap of the probability for the correct class between the original sample and the transformed sample. The ratio is described as the ratio of the prediction confidence from the original sample over the prediction confidence from the transformed sample is greater than e.

Validation of **A6**:

- We inspect Assumption **A6** by directly calculating the frequencies out of all the samples when it holds. Given a vanilla model (Base), we notice that over 74% samples out of 50000 samples fit this assumption. Thus, we consider this assumption reasonable enough to use.

A1.2 Proof of Theoretical Results

Lemma A1.1. *With Assumptions A1, A4, and A5, with probability at least* $1-\delta$*, we have*

$$r_{\mathbf{P},\mathcal{A}}(\widehat{\theta}) \leqslant \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}} \sup_{a\in\mathcal{A}} \mathbb{I}(g(f(a(\mathbf{x});\widehat{\theta})) \neq \mathbf{y}) + \phi(|\Theta|, n, \delta)$$

Proof. With Assumption A5, we simply say

$$\arg\max_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} r_{\mathbf{P}'}(\widehat{\theta}) = \arg\max_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) = \mathbf{P}_w$$

we can simply analyze the expected risk following the standard classical techniques since both expected risk and empirical risk are studied over distribution $\mathbf{P}_w$.

Now we only need to make sure the classical analyses (as discussed in A4) are still valid over distribution $\mathbf{P}_w$:

- when **A4** is "Θ is finite, $l(\cdot,\cdot)$ is a zero-one loss, samples are *i.i.d*", $\phi(|\Theta|, n, \delta) = \sqrt{\dfrac{\log(|\Theta|) + \log(1/\delta)}{2n}}$. The proof of this result uses Hoeffding's inequality, which only requires independence of random variables. One can refer to Section 3.6 in [97] for the detailed proof.
- when **A4** is "samples are *i.i.d*", $\phi(|\Theta|, n, \delta) = 2\mathcal{R}(\mathcal{L}) + \sqrt{\dfrac{\log 1/\delta}{2n}}$. The proof of this result relies on McDiarmid's inequality, which also only requires independence of random variables. One can refer to Section 3.8 in [97] for the detailed proof.

Assumption **A1** guarantees the samples from distribution $\mathbf{P}_w$ are still independent, thus the generic term holds for at least these two concrete examples, thus the claim is proved.

□

Proposition A1.2. *With A2, for any* $a \in \mathcal{A}$*, we have*

$$W_1(\widehat{\mathbf{Q}}_{\mathbf{x},\widehat{\theta}}, \widehat{\mathbf{Q}}_{a(\mathbf{x}),\widehat{\theta}}) = \sum_{i}^{|(\mathbf{X},\mathbf{Y})|} ||f(\mathbf{x}_i;\widehat{\theta}) - f(a(\mathbf{x}_i);\widehat{\theta})||_1,$$

where $\widehat{\mathbf{Q}}_{\mathbf{x},\widehat{\theta}}$ *denotes the empirical distribution of* $f(a(\mathbf{x});\widehat{\theta})$ *for* $(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})$.

Proof. We leverage the order statistics representation of Wasserstein metric over empirical distributions (*e.g.*, see Section 4 in [11])

$$W_1(\widehat{\mathbf{Q}}_{\mathbf{x},\widehat{\theta}}, \widehat{\mathbf{Q}}_{a(\mathbf{x}),\widehat{\theta}})) = \inf_{\sigma} \sum_{i}^{|(\mathbf{X},\mathbf{Y})|} ||f(\mathbf{x}_i;\widehat{\theta}) - f(a(\mathbf{x}_{\sigma(i)}),\widehat{\theta})||_1$$

where σ stands for a permutation of the index, thus the infimum is taken over all possible permutations. With Assumption A2, when $d_e(\cdot,\cdot)$ in A2 chosen to be ℓ_1 norm, we have:

$$||f(\mathbf{x}_i;\widehat{\theta}) - f(a(\mathbf{x}_i);\widehat{\theta})||_1 \leqslant \min_{j\neq i} ||f(\mathbf{x}_i;\widehat{\theta}) - f(a(\mathbf{x}_j),\widehat{\theta})||_1$$

Thus, the infimum is taken when σ is the natural order of the samples, which leads to the claim.

□

Theorem. *With Assumptions A1, A2, A4, A5, and A6, with probability at least* $1-\delta$*, we have*

$$r_{\mathbf{P},\mathcal{A}}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + \sum_i ||f(\mathbf{x}_i;\widehat{\theta}) - f(\mathbf{x}_i';\widehat{\theta})||_1 + \phi(|\Theta|,n,\delta) \tag{A.4}$$

and $\mathbf{x}' = a(\mathbf{x})$*, where* $a = \arg\max_{a\in\mathcal{A}} \mathbf{y}^\top f(a(\mathbf{x});\widehat{\theta})$*.*

Proof. First of all, in the context of multiclass classification, where $g(f(\mathbf{x},;\theta))$ predicts a label with one-hot representation, and $\mathbf{y}$ is also represented with one-hot representation, we can have the empirical risk written as:

$$\widehat{r}_{\mathbf{P}}(\widehat{\theta}) = 1 - \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}} \mathbf{y}^\top g(f(\mathbf{x};\widehat{\theta}))$$

Thus,

$$\begin{aligned}\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) &= \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + \sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) - \widehat{r}_{\mathbf{P}}(\widehat{\theta}) \\ &= \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + \frac{1}{n}\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \Big(\sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}} \mathbf{y}^\top g(f(\mathbf{x};\widehat{\theta})) - \sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}'} \mathbf{y}^\top g(f(\mathbf{x};\widehat{\theta}))\Big)\end{aligned}$$

With A6, we can continue with:

$$\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + \frac{1}{n}\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \Big(\sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}} \mathbf{y}^\top \log(f(\mathbf{x};\widehat{\theta})) - \sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}'} \mathbf{y}^\top \log(f(\mathbf{x};\widehat{\theta}))\Big)$$

If we use $e(\cdot) = -\mathbf{y}^\top\log(\cdot)$ to replace the cross-entropy loss, we simply have:

$$\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + \frac{1}{n}\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \Big(\sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}'} e(f(\mathbf{x};\widehat{\theta})) - \sum_{(\mathbf{x},\mathbf{y})\sim\mathbf{P}} e((f(\mathbf{x};\widehat{\theta}))\Big)$$

Since $e(\cdot)$ is a Lipschitz function with constant $\leqslant 1$ (because of A6, Eq.(A.3)) and together with the dual representation of Wasserstein metric (See *e.g.*, [156]), we have

$$\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + W_1(\widehat{\mathbf{Q}}_{\mathbf{x},\hat{\theta}}, \widehat{\mathbf{Q}}_{a(\mathbf{x}),\hat{\theta}}))$$

where $\mathbf{x}' = a(\mathbf{x})$, where $a = \arg\max_{a\in\mathcal{A}} \mathbf{y}^\top f(a(\mathbf{x});\widehat{\theta})$; $\widehat{\mathbf{Q}}_{\mathbf{x},\hat{\theta}}$ denotes the empirical distribution of $f(a(\mathbf{x});\widehat{\theta})$ for $(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})$. Note that $r_{\mathbf{P},\mathcal{A}}(\widehat{\theta})$, by definition, is a shorthand notation for $\sup_{\mathbf{P}'\in T(\mathbf{P},\mathcal{A})} r_{\mathbf{P}'}(\widehat{\theta})$.

Further, we can use the help of Proposition B.2 to replace Wassertein metric with ℓ_1 distance. Finally, we can conclude the proof with Assumption A5 as how we did in the proof of Lemma B.1. □

Lemma. *With Assumptions A1-A6, assuming there is a* $a'() \in \mathcal{A}$ *where* $\widehat{r}_{\mathbf{P}_{a'}}(\widehat{\theta}) = \frac{1}{2}\big(\widehat{r}_{\mathbf{P}_{a^+}}(\widehat{\theta}) + \widehat{r}_{\mathbf{P}_{a^-}}(\widehat{\theta})\big)$*, with probability at least* $1-\delta$*, we have:*

$$r_{\mathbf{P},\mathcal{A}}(\widehat{\theta}) \leqslant \frac{1}{2}\big(\widehat{r}_{\mathbf{P}_{a^+}}(\widehat{\theta}) + \widehat{r}_{\mathbf{P}_{a^-}}(\widehat{\theta})\big) + \sum_i ||f(a^+(\mathbf{x}_i);\widehat{\theta}) - f(a^-(\mathbf{x}');\widehat{\theta})||_1 + \phi(|\Theta|,n,\delta) \tag{A.5}$$

Proof. We can continue with

$$\sup_{\mathbf{P}' \in T(\mathbf{P}, \mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + W_1(\widehat{\mathbf{Q}}_{\mathbf{x},\widehat{\theta}}, \widehat{\mathbf{Q}}_{a(\mathbf{x}),\widehat{\theta}})),$$

where $\widehat{\mathbf{Q}}_{\mathbf{x},\widehat{\theta}}$ denotes the empirical distribution of $f(a(\mathbf{x}); \widehat{\theta})$ for $(\mathbf{x}, \mathbf{y}) \in (\mathbf{X}, \mathbf{Y})$. from the proof of Theorem 5.2. With the help of Assumption A3, we have:

$$d_x(f(a^+(\mathbf{x}), \widehat{\theta}), f(a^-(\mathbf{x}), \widehat{\theta})) \geqslant d_x(f(\mathbf{x}, \widehat{\theta}), f(\mathbf{x}', \widehat{\theta}))$$

When $d_x(\cdot, \cdot)$ is chosen as Wasserstein-1 metric, we have:

$$\sup_{\mathbf{P}' \in T(\mathbf{P}, \mathcal{A})} \widehat{r}_{\mathbf{P}'}(\widehat{\theta}) \leqslant \widehat{r}_{\mathbf{P}}(\widehat{\theta}) + W_1(\widehat{\mathbf{Q}}_{a^+(\mathbf{x}),\widehat{\theta}}, \widehat{\mathbf{Q}}_{a^-(\mathbf{x}),\widehat{\theta}}))$$

Further, as the LHS is the worst case risk generated by the transformation functions within $\mathcal{A}$, and $\widehat{r}_{\mathbf{P}}(\widehat{\theta})$ is independent of the term $W_1(\widehat{\mathbf{Q}}_{a^+(\mathbf{x}),\widehat{\theta}}, \widehat{\mathbf{Q}}_{a^-(\mathbf{x}),\widehat{\theta}}))$, WLOG, we can replace $\widehat{r}_{\mathbf{P}}(\widehat{\theta})$ with the risk of an arbitrary distribution generated by the transformation function in $\mathcal{A}$. If we choose to use $\widehat{r}_{\mathbf{P}_{a'}}(\widehat{\theta}) = \frac{1}{2}\big(\widehat{r}_{\mathbf{P}_{a^+}}(\widehat{\theta}) + \widehat{r}_{\mathbf{P}_{a^-}}(\widehat{\theta})\big)$, we can conclude the proof with help from Proposition B.2 and Assumption A5 as how we did in the proof of Theorem 5.2. □

Table A3: Details of Rotation Experiment

	ResNet			ResNet-GC			ResNet-ST			ResNet-ETN		
	Base	RVA	RWA	Base	RVA	RWA	Base	RVA	RWA	Base	RVA	RWA
0	0.836	0.8487	0.8708	0.7645	0.8537	0.8578	0.7805	0.7472	0.7787	0.8242	0.8537	0.8562
15	0.6938	0.7904	0.8871	0.5596	0.793	0.8538	0.7374	0.7361	0.7706	0.7151	0.8199	0.8538
30	0.4557	0.7455	0.8869	0.3558	0.7485	0.8275	0.6519	0.7362	0.7715	0.4407	0.8381	0.8467
45	0.3281	0.8005	0.887	0.246	0.7618	0.7482	0.4953	0.7375	0.7727	0.2839	0.8401	0.835
60	0.2578	0.8282	0.8818	0.1963	0.8251	0.8191	0.3974	0.7374	0.7663	0.2297	0.845	0.8395
75	0.2366	0.7236	0.8101	0.1878	0.7976	0.6715	0.3177	0.7342	0.7374	0.2484	0.8051	0.5985
90	0.2939	0.5615	0.5742	0.1966	0.6589	0.617	0.3044	0.7235	0.6721	0.2923	0.4452	0.3494
105	0.2027	0.3545	0.4242	0.1717	0.4483	0.5748	0.2698	0.7223	0.649	0.2225	0.4245	0.3059
120	0.1758	0.2992	0.3885	0.1651	0.4162	0.5996	0.2677	0.7169	0.6663	0.1877	0.3868	0.3017
135	0.1748	0.3115	0.3708	0.1683	0.3854	0.4152	0.2655	0.7145	0.6585	0.1907	0.4029	0.34
150	0.181	0.3347	0.3524	0.184	0.3777	0.3631	0.2813	0.7097	0.6266	0.2082	0.4012	0.2992
165	0.2283	0.325	0.3366	0.2091	0.3604	0.3174	0.3091	0.7054	0.5611	0.2585	0.3618	0.2953
180	0.3053	0.3485	0.3795	0.2673	0.3669	0.3183	0.3295	0.7176	0.534	0.3343	0.3796	0.3577
195	0.2607	0.3089	0.3781	0.2265	0.3221	0.3256	0.2985	0.705	0.5177	0.2663	0.3543	0.3445
210	0.2298	0.3109	0.3806	0.1963	0.3258	0.3468	0.2803	0.7003	0.5173	0.2225	0.362	0.3447
225	0.2218	0.3342	0.3723	0.1788	0.3316	0.3315	0.2687	0.6965	0.5151	0.1948	0.3622	0.3468
240	0.2042	0.3519	0.3729	0.1755	0.3613	0.3808	0.2558	0.7033	0.5235	0.1894	0.3597	0.3332
255	0.2023	0.3335	0.3631	0.194	0.36	0.3964	0.2663	0.7147	0.5597	0.2206	0.3706	0.2964
270	0.2683	0.3507	0.381	0.2297	0.4607	0.4411	0.3202	0.7318	0.6356	0.291	0.4945	0.3372
285	0.2275	0.3046	0.389	0.2056	0.5844	0.4521	0.335	0.7245	0.6339	0.2527	0.4255	0.3249
300	0.2196	0.3198	0.4012	0.2117	0.6469	0.4973	0.3686	0.7232	0.6443	0.2393	0.4292	0.3315
315	0.2573	0.3901	0.4251	0.2427	0.5596	0.4816	0.4211	0.7303	0.6412	0.2746	0.4588	0.3259
330	0.3873	0.5489	0.4852	0.3429	0.5755	0.7211	0.5552	0.7299	0.6592	0.4215	0.5057	0.3534
345	0.6502	0.717	0.6765	0.5463	0.74	0.8417	0.7193	0.7379	0.7215	0.7055	0.7023	0.6528

A1.3 Additional Results for Comparisons with Advanced Methods

We have also conducted two full ImageNet level experiments. However, due to the limitation of resources, we cannot tune the models substantially. Our current trial suggest that our techniques can improve the vanilla model to compete with SOTA models, limited by our resources, we cannot do wide-range hyperparameters search to outperform them. Also, considering the fact that many of these methods are significantly more complicated than us and also uses data augmentation specially designed for the tasks, we consider our experiments a success indication of the empirical strength of our methods.

Texture-perturbed ImageNet classification We also test the performance on the image classification over multiple perturbations. We train the model over standard ImageNet training set and test the model with ImageNet-C data [59], which is a perturbed version of ImageNet by corrupting the original ImageNet validation set with a collection of noises. Following the standard, the reported performance is mCE, which is the smaller the better. We compare with several methods tested on this dataset, including Patch Uniform (PU) [102], AutoAugment (AA) [23], MaxBlur pool (MBP) [183], Stylized ImageNet (SIN) [59], AugMix (AM) [62], AugMix w. SIN (AMS) [62]. We use the performance reported in [62]. Again, our augmention only uses the generic texture with perturbation (the $\mathcal{A}$ in our texture synthetic experiments with radius changed to $20, 25, 30, 35, 40$). The results are reported in Table A4, which shows that our generic method outperform the current SOTA methods after a continued finetuning process with reducing learning rates.

Cross-domain ImageNet-Sketch Classification We also compare to the methods used for cross-domain evaluation. We follow the set-up advocated by [163] for domain-agnostic cross-domain prediction, which is training the model on one or multiple domains without domain identifiers and test the model on an unseen domain. We use the most challenging setup in this scenario: train the models with standard ImageNet training data, and test the model over

	Clean	Noise			Blur				Weather				Digital				mCE
		Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	
Base	23.9	79	80	82	82	90	84	80	86	81	75	65	79	91	77	80	80.6
RVA	23.6	78	78	79	74	87	79	76	78	75	69	58	68	85	75	75	75.6
RWA	22.4	61	63	63	68	75	65	66	70	69	64	56	55	70	61	63	64.6
SU	24.5	67	68	70	74	83	81	77	80	74	75	62	77	84	71	71	74.3
AA	22.8	69	68	72	77	83	80	81	79	75	64	56	70	88	57	71	72.7
MBP	23	73	74	76	74	86	78	77	77	72	63	56	68	86	71	71	73.4
SIN	27.2	69	70	70	77	84	76	82	74	75	69	65	69	80	64	77	73.3
AM	22.4	65	66	67	70	80	66	66	75	72	67	58	58	79	69	69	68.4
AMS	25.2	61	62	61	69	77	63	72	66	68	63	59	52	74	60	67	64.9

Table A4: Comparison to advanced models over ImageNet-C data. Performance reported (mCE) follows the standard in ImageNet-C data: mCE is the smaller the better.

	Base	InfoDrop	HEX	PAR	VA	RVA	RSC	VWA	RWA
Top-1	0.1204	0.1224	0.1292	0.1306	0.1362	0.1405	0.1612	0.1432	0.1486
Top-5	0.2408	0.256	0.2564	0.2627	0.2715	0.2793	0.3078	0.2846	0.2933

Table A5: Comparison to advanced cross-domain image classification models, over ImageNet-Sketch dataset. We report top-1 and top-5 accuracy following standards on ImageNet related experiments.

ImageNet-Sketch data [162], which is a collection of sketches following the structure ImageNet validation set. We compare with previous methods with reported performance on this dataset, such as InfoDrop [1], HEX [163], PAR [162], RSC [69] and report the performances in Table A5. Notice that, our data augmentation also follows the requirement that the characteristics of the test domain cannot be utilized during training. Thus, we only augment the samples with a generic augmentation set ($\mathcal{A}$ of "contrast" in synthetic experiments). The results again support the usage of data augmentation and consistency loss.

A2 Proofs for Chapter 4.2: Generalization Bound of Learning Robust Models

A2.1 Lemma A3.1 and Proof

Lemma A2.1. *With sample* $(\mathbf{x}, \mathbf{y})$ *and two labeling functions* $f_1(\mathbf{x}) = f_2(\mathbf{x}) = \mathbf{y}$*, for an estimated* $\theta \in \Theta$*, if* $\theta(\mathbf{x}) = \mathbf{y}$*, then with* ***A3*** *, we have*

$$d(\theta, f_1, \mathbf{x}) = 1 \implies r(\theta, \mathcal{A}(f_2, \mathbf{x})) = 1. \tag{A.6}$$

Proof. If $\theta(\mathbf{x}) = \mathbf{y}$ and $d(\theta, f_1, \mathbf{x}) = 1$, according to **A3**, we have $d(\theta, f_2, \mathbf{x}) = 0$.

We prove this by contradiction.

If the conclusion does not hold, $r(\theta, \mathcal{A}(f_2, \mathbf{x})) = 0$, which means

$$\max_{\mathbf{x}_{\mathcal{A}(f_2,\mathbf{x})} \in \mathcal{X}_{\mathcal{A}(f_2,\mathbf{x})}} |\theta(\mathbf{x}) - \mathbf{y}| = 0 \tag{A.7}$$

Together with $d(\theta, f_2, \mathbf{x}) = 0$, which means

$$\max_{\mathbf{z}\in\mathcal{X}:\mathbf{z}_{\mathcal{A}(f_2,\mathbf{x})}=\mathbf{x}_{\mathcal{A}(f_2,\mathbf{x})}} |\theta(\mathbf{z}) - \mathbf{y}| = 0, \tag{A.8}$$

we will have

$$\max_{\mathbf{x}\in\mathcal{X}} |\theta(\mathbf{x}) - \mathbf{y}| = 0, \tag{A.9}$$

which is $\theta(\mathbf{x}) = \mathbf{y}$ for any $\mathbf{x} \in \mathbf{P}$.

This contradicts with the premises in **A3** (θ is not a constant function).

□

A2.2 Theorem 3.1 and Proof

Theorem. *With Assumptions **A1**-**A3**, with probability as least $1 - \delta$, we have*

$$\epsilon_{\mathbf{P}_t}(\theta) \leqslant \widehat{\epsilon}_{\mathbf{P}_s}(\theta) + c(\theta) + \phi(|\Theta|, n, \delta) \tag{A.10}$$

where $c(\theta) = \dfrac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] r(\theta, \mathcal{A}(f_m, \mathbf{x}))$.

Proof.

$$\widehat{\epsilon}_{\mathbf{P}_s}(\theta) = \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} |\theta(\mathbf{x}) - f(\mathbf{x})| \tag{A.11}$$

$$= 1 - \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \big(\mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})]\big) \tag{A.12}$$

$$= 1 - \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \big(\mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})]\mathbb{I}[d(\theta, f_h, \mathbf{x}) = 0] + \mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})]\mathbb{I}[d(\theta, f_h, \mathbf{x}) = 1]\big) \tag{A.13}$$

$$= 1 - \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \big(\mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})]\mathbb{I}[d(\theta, f_h, \mathbf{x}) = 0]\big) - \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})]\mathbb{I}[d(\theta, f_h, \mathbf{x}) = \tag{A.14}$$

$$\geqslant \widehat{\epsilon}_d(\theta) - \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})] r(\theta, \mathcal{A}(f_m, \mathbf{x})), \tag{A.15}$$

where the last line used Lemma A2.1.

Thus, we have

$$\widehat{\epsilon}_d(\theta) \leqslant \widehat{\epsilon}(\theta) + \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})] r(\theta, \mathcal{A}(f_m, \mathbf{x})) \tag{A.16}$$

where

$$\widehat{\epsilon}_d(\theta) = 1 - \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \big(\mathbb{I}[\theta(\mathbf{x}) = f(\mathbf{x})]\mathbb{I}[d(\theta, f_h, \mathbf{x}) = 0]\big), \tag{A.17}$$

which describes the correctly predicted terms that θ functions the same as f_h and all the wrongly predicted terms. Therefore, conventional generalization analysis through uniform convergence applies, and we have

$$\epsilon_{\mathbf{P}_t}(\theta) \leqslant \widehat{\epsilon}_d(\theta) + \phi(|\Theta|, n, \delta) \tag{A.18}$$

Thus, we have:

$$\epsilon_{\mathbf{P}_t}(\theta) \leqslant \widehat{\epsilon}_{\mathbf{P}_s}(\theta) + \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]r(\theta, \mathcal{A}(f_m, \mathbf{x})) + \phi(|\Theta|, n, \delta) \tag{A.19}$$

□

A2.3 Theorem 3.2 and Proof

Theorem. *With Assumptions **A2-A4**, and if* $1 - f_h \in \Theta$*, we have*

$$c(\theta) \leqslant D_\Theta(\mathbf{P}_s, \mathbf{P}_t) + \frac{1}{n} \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]r(\theta, \mathcal{A}(f_m, \mathbf{x})) \tag{A.20}$$

where $c(\theta) = \frac{1}{n}\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]r(\theta, \mathcal{A}(f_m, \mathbf{x}))$ *and* $D_\Theta(\mathbf{P}_s, \mathbf{P}_t)$ *is defined as in* (4.8)*.*

Proof. By definition, $g(\mathbf{x}) \in \Theta\Delta\Theta \iff g(\mathbf{x}) = \theta(\mathbf{x}) \oplus \theta'(\mathbf{x})$ for some $\theta, \theta' \in \Theta$, together with

Lemma 2 and Lemma 3 of [9], we have

$$D_\Theta(\mathbf{P}_s, \mathbf{P}_t) = \frac{1}{n} \max_{\theta,\theta' \in \Theta} \Big| \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} |\theta(\mathbf{x}) - \theta'(\mathbf{x})| - \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} |\theta(\mathbf{x}) - \theta'(\mathbf{x})| \Big| \tag{A.21}$$

$$\geqslant \frac{1}{n} \Big| \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} |\theta(\mathbf{x}) - f_z(\mathbf{x})| - \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} |\theta(\mathbf{x}) - f_z(\mathbf{x})| \Big| \tag{A.22}$$

$$= \frac{1}{n} \Big| \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] - \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \Big| \tag{A.23}$$

$$= \frac{1}{n} \Big| \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 1] - \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 1$$

(A.24)

$$+ \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 0] - \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 0$$

(A.25)

$$= \frac{1}{n} \Big| \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] r(\theta, \mathcal{A}(f_m, \mathbf{x})) - \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] r(\theta, \mathcal{A}(f_m, \mathbf{x})) \Big|$$

(A.26)

$$\geqslant c(\theta) - \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] r(\theta, \mathcal{A}(f_m, \mathbf{x})) \tag{A.27}$$

First line: see Lemma 2 and Lemma 3 of [9].

Second line: if $1 - f_h \in \Theta$, and we use f_z to denote $1 - f_h$.

Fifth line is a result of using that fact that

$$\sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 0] = \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 0] \tag{A.28}$$

as a result of our assumptions. Now we present the details of this argument:

According to **A3**, if $\theta(\mathbf{x}) = \mathbf{y}$, $d(\theta, f_h, \mathbf{x}) d(\theta, f_m, \mathbf{x}) = 0$. Since $r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 0$, $d(\theta, f_m, \mathbf{x})$ cannot be 0 unless θ is a constant mapping that maps every sample to 0 (which will contradicts **A3**). Thus, we have $d(\theta, f_h, \mathbf{x}) = 0$.

Therefore, we can rewrite the left-hand term following

$$\sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 0] = \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[d(\theta, f_h, \mathbf{x}) = 0] \tag{A.29}$$

and similarly

$$\sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[r(\theta, \mathcal{A}(f_m, \mathbf{x})) = 0] = \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}] \mathbb{I}[d(\theta, f_h, \mathbf{x}) = 0] \tag{A.30}$$

We recap the definition of $d_x(\cdot,\cdot)$, thus $d(\theta, f_h, \mathbf{x}) = 0$ means

$$d(\theta, f_h, \mathbf{x}) = \max_{\mathbf{z}\in\mathcal{X}:\mathbf{z}_{\mathcal{A}(f,\mathbf{x})}=\mathbf{x}_{\mathcal{A}(f_h,\mathbf{x})}} |\theta(\mathbf{z}) - f_h(\mathbf{z})| = 0 \tag{A.31}$$

Therefore $d(\theta, f_h, \mathbf{x}) = 0$ implies $\mathbb{I}(\theta(\mathbf{x}) = \mathbf{y})$, and

$$|\theta(\mathbf{z}) - f_h(\mathbf{z})| = 0 \quad \forall \quad \mathbf{z}_{\mathcal{A}(f_h,\mathbf{x})} = \mathbf{x}_{\mathcal{A}(f_h,\mathbf{x})} \tag{A.32}$$

Therefore, we can continue to rewrite the left-hand term following

$$\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]\mathbb{I}[d(\theta, f_h, \mathbf{x}) = 0] = \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{z}) - f_h(\mathbf{z})] = \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) - f_h(\mathbf{x})] \tag{A.33}$$

and similarly

$$\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{x}) = \mathbf{y}]\mathbb{I}[d(\theta, f_h, \mathbf{x}) = 0] = \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{z}) - f_h(\mathbf{z})] \tag{A.34}$$

where $\mathbf{z}$ denotes any $\mathbf{z} \in \mathcal{X}$ and $\mathbf{z}_{\mathcal{A}(f_h,\mathbf{x})} = \mathbf{x}_{\mathcal{A}(f_h,\mathbf{x})}$.

Further, because of **A4**, we have

$$\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_t}} \mathbb{I}[\theta(\mathbf{z}) - f_h(\mathbf{z})] = \sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})_{\mathbf{P}_s}} \mathbb{I}[\theta(\mathbf{x}) - f_h(\mathbf{x})]. \tag{A.35}$$

Thus, we showed the (A.28) holds and conclude our proof.

□

A3 Proofs for Chapter 4.4: The Self-Challenging Algorithm

A3.1 Corollary 1

Proof. We first study the convergence part, where we consider a fixed hypothesis. We first expand

$$\begin{aligned}
&|L(\widehat{\theta}_{\text{RSC}}(\mathcal{S});\mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{D})| \\
&= |L(\widehat{\theta}_{\text{RSC}}(\mathcal{S});\mathcal{S}) - L(\widehat{\theta}_{\text{RSC}}(\mathcal{S});\mathcal{D}) + L(\widehat{\theta}_{\text{RSC}}(\mathcal{S});\mathcal{D}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{D})| \\
&\leqslant |L(\widehat{\theta}_{\text{RSC}}(\mathcal{S});\mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{D})| + |L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{D}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{D})|
\end{aligned}$$

We first consider the term $|L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{D})|$, where we can expand

$$|L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{D})| \leqslant 2|L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S});\mathcal{O})|$$

because of Assumption **A4**.

Also, because of Assumption **A4**, if samples in $\mathcal{S}$ are perturbed versions of samples in $\mathcal{O}$, then samples in $\mathcal{O}$ can also be seen as perturbed versions of samples in $\mathcal{S}$, thus, Condition 4.21 can be directly re-written into:

$$|L(\theta^{\star}_{\text{RSC}}(\mathcal{S}); \mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S}); \mathcal{O})| \leqslant \xi(p),$$

which directly leads us to the fact that $|L(\theta^{\star}_{\text{RSC}}(\mathcal{S}); \mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S}); \mathcal{D})|$ has the expectation 0 (**A4**) and bounded by $[0, \xi(p)]$.

For $|L(\widehat{\theta}_{\text{RSC}}(\mathcal{S}); \mathcal{S}) - L(\theta^{\star}_{\text{RSC}}(\mathcal{S}); \mathcal{S})|$, the strategy is relatively standard. We first consider the convergence of a fixed hypothesis θ_{RSC}, then over n *i.i.d* samples, the empirical risk ($\widehat{L}(\theta_{\text{RSC}})$) will be bounded within $[0, 1]$ with the expectation $L(\theta_{\text{RSC}})$.

Before we consider the uniform convergence step, we first put the two terms together and apply the Hoeffding's inequality. When the random variable is with expectation $L(\theta_{\text{RSC}})$ and bound $[0, 1 + 2\xi(p)]$, we have:

$$\mathbb{P}(|\widehat{L}(\theta_{\text{RSC}}; \mathcal{S}) - L(\theta_{\text{RSC}}; \mathcal{D})| \geqslant \epsilon) \leqslant 2\exp(-\frac{2n\epsilon^2}{(2\xi(p)+1)^2})$$

Now, we consider the uniform convergence case, where we have:

$$\mathbb{P}(\sup_{\theta_{\text{RSC}} \in \Theta_{\text{RSC}}} |\widehat{L}(\theta_{\text{RSC}}; \mathcal{S}) - L(\theta_{\text{RSC}}; \mathcal{D})| \geqslant \epsilon) \leqslant 2|\Theta_{\text{RSC}}| \exp(-\frac{2n\epsilon^2}{(2\xi(p)+1)^2})$$

Rearranging these terms following standard tricks will lead to the conclusion.

□

A3.2 Corollary 2

Proof. Since we only concern with iteration t, we drop the subscript of $\mathbf{z}_t$ and $\tilde{\mathbf{z}}_t$. We first introduce another shorthand notation

$$h(\widehat{\theta}_{\text{RSC}}(t+1), \mathbf{z}) := \sum_{\langle \mathbf{z}_t, \mathbf{y} \rangle} l(f(\mathbf{z}; \widehat{\theta}_{\text{RSC}}); \mathbf{y})$$

We expand

$$\begin{aligned}
\Gamma(\widehat{\theta}_{\text{RSC}}(t+1)) =& |h(\widehat{\theta}_{\text{RSC}}(t+1), \mathbf{z}) - h(\widehat{\theta}_{\text{RSC}}(t+1), \tilde{\mathbf{z}})| \\
=& |h(\widehat{\theta}_{\text{RSC}}(t+1), \mathbf{z}) - h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}) + h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}) - h(\widehat{\theta}_{\text{RSC}}(t+1), \tilde{\mathbf{z}})| \\
=& |h(\widehat{\theta}_{\text{RSC}}(t+1), \mathbf{z}) - h(\widehat{\theta}_{\text{RSC}}(t), \mathbf{z}) + h(\widehat{\theta}_{\text{RSC}}(t), \mathbf{z}) - h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}) \\
& + h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}) - h(\widehat{\theta}_{\text{RSC}}(t+1), \tilde{\mathbf{z}})| \\
=& |h(\widehat{\theta}_{\text{RSC}}(t+1), \mathbf{z}) - h(\widehat{\theta}_{\text{RSC}}(t), \mathbf{z}) + h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}}) - h(\widehat{\theta}_{\text{RSC}}(t+1), \tilde{\mathbf{z}}) + \Gamma(\widehat{\theta}_{\text{RSC}}(t))|
\end{aligned}$$

Recall that, by the definition of RSC, we have:

$$\widehat{\theta}_{\text{RSC}}(t+1) = \widehat{\theta}_{\text{RSC}}(t) - \frac{\partial h(\widehat{\theta}_{\text{RSC}}(t), \tilde{\mathbf{z}})}{\partial \widehat{\theta}_{\text{RSC}}(t)} \eta = \widehat{\theta}_{\text{RSC}}(t) - \tilde{\mathbf{g}}\eta$$

We apply Taylor expansion over $h(\widehat{\theta}_{\text{RSC}}(t+1),\cdot)$ with respect to $\widehat{\theta}_{\text{RSC}}(t)$ and have:

$$\begin{aligned}h(\widehat{\theta}_{\text{RSC}}(t+1),\cdot) =& h(\widehat{\theta}_{\text{RSC}}(t),\cdot) + \frac{\partial h(\widehat{\theta}_{\text{RSC}}(t),\cdot)}{\partial \widehat{\theta}_{\text{RSC}}(t)}(\widehat{\theta}_{\text{RSC}}(t+1) - \widehat{\theta}_{\text{RSC}}(t)) \\ &+ \frac{1}{2}\frac{\partial^2 h(\widehat{\theta}_{\text{RSC}}(t),\cdot)}{\partial^2 \widehat{\theta}_{\text{RSC}}(t)}||\widehat{\theta}_{\text{RSC}}(t+1) - \widehat{\theta}_{\text{RSC}}(t)||_2^2 + \sigma \\ =& h(\widehat{\theta}_{\text{RSC}}(t),\cdot) - \frac{\partial h(\widehat{\theta}_{\text{RSC}}(t),\cdot)}{\partial \widehat{\theta}_{\text{RSC}}(t)}\tilde{\mathbf{g}}\eta + \frac{1}{2}\frac{\partial^2 h(\widehat{\theta}_{\text{RSC}}(t),\cdot)}{\partial^2 \widehat{\theta}_{\text{RSC}}(t)}||\tilde{\mathbf{g}}\eta||_2^2 + \sigma,\end{aligned}$$

where σ denotes the higher order terms.

Assumption **A6** conveniently allows us to drop terms regarding η^2 or higher orders, so we have:

$$h(\widehat{\theta}_{\text{RSC}}(t),\cdot) - h(\widehat{\theta}_{\text{RSC}}(t+1),\cdot) = \frac{\partial h(\widehat{\theta}_{\text{RSC}}(t),\cdot)}{\partial \widehat{\theta}_{\text{RSC}}(t)}\tilde{\mathbf{g}}\eta \tag{A.36}$$

Finally, when $\cdot$ is replaced by $\mathbf{z}$ and $\tilde{\mathbf{z}}$,

we have:

$$h(\widehat{\theta}_{\text{RSC}}(t),\tilde{\mathbf{z}}) - h(\widehat{\theta}_{\text{RSC}}(t+1),\tilde{\mathbf{z}}) = \frac{\partial h(\widehat{\theta}_{\text{RSC}}(t),\tilde{\mathbf{z}})}{\partial \widehat{\theta}_{\text{RSC}}(t)}\tilde{\mathbf{g}}\eta = ||\tilde{\mathbf{g}}||_2^2\eta$$

and

$$h(\widehat{\theta}_{\text{RSC}}(t),\mathbf{z}) - h(\widehat{\theta}_{\text{RSC}}(t+1),\mathbf{z}) = \frac{1}{\gamma_t(p)}\frac{\partial h(\widehat{\theta}_{\text{RSC}}(t),\tilde{\mathbf{z}})}{\partial \widehat{\theta}_{\text{RSC}}(t)}\tilde{\mathbf{g}}\eta = \frac{1}{\gamma_t(p)}||\tilde{\mathbf{g}}||_2^2\eta$$

We write these terms back and get

$$\Gamma(\widehat{\theta}_{\text{RSC}}(t+1)) = \left|(\frac{1}{\gamma_t(p)} - 1)||\tilde{\mathbf{g}}||_2^2\eta + \Gamma(\widehat{\theta}_{\text{RSC}}(t))\right|$$

We can simply drop the absolute value sign because all these terms are greater than zero. Finally, we rearrange these terms and prove the conclusion.

□

Bibliography

[1] A. Achille and S. Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (12):2897–2905, 2018. 3.3.2, 3.3.3, A1.3

[2] D. Aneja, A. Colburn, G. Faigin, L. Shapiro, and B. Mones. Modeling stylized character expressions via deep learning. In *Asian Conference on Computer Vision*, pages 136–153. Springer, 2016. 3.3.2

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017. 3.2.2, 3.2.4

[4] A. Asai and H. Hajishirzi. Logic-guided data augmentation and regularization for consistent question answering, 2020. 3.2.2

[5] H. Bahng, S. Chun, S. Yun, J. Choo, and S. J. Oh. Learning de-biased representations with biased representations. *arXiv preprint arXiv:1910.02806*, 2019. 1.1, 3.2.6

[6] Y. Balaji, S. Sankaranarayanan, and R. Chellappa. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pages 998–1008, 2018. 3.3.3, 4.4.1, 4.4.3, 4.4.3

[7] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. 3.3.2

[8] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007. 1.1, 3.1, 4.1, 4.2.1

[9] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. 1.1, 3.1, 3.2.1, 3.2.3, 3.3.1, 4.1, 4.2.1, 4.2.1, 4.2.1, 4.4.1, 4.4.2, A2.3, A2.3

[10] S. Ben-David, T. Lu, T. Luu, and D. Pál. Impossibility theorems for domain adaptation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. 3.3.1

[11] S. Bobkov and M. Ledoux. *One-dimensional empirical measures, order statistics, and Kantorovich transport distances*, volume 261. American Mathematical Society, 2019. A1.2

[12] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016. 3.3.2, 3.3.3

[13] O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In

Summer School on Machine Learning, pages 169–207. Springer, 2003. 4.1, A1.1

[14] R. N. Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986. 2.2.3

[15] J. S. Bridle and S. J. Cox. Recnorm: Simultaneous normalisation and classification applied to speech recognition. In *Advances in Neural Information Processing Systems*, pages 234–240, 1991. 3.3.1, 4.4.1

[16] R. Cadene, C. Dancette, M. Cord, D. Parikh, et al. Rubi: Reducing unimodal biases for visual question answering. In *Advances in neural information processing systems*, pages 841–852, 2019. 3.2.6

[17] F. M. Carlucci, P. Russo, T. Tommasi, and B. Caputo. Agnostic domain generalization. *arXiv preprint arXiv:1808.01102*, 2018. 3.3.1

[18] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019. 3.3.3, 3.11, 3.3.3, 4.4.1, 4.4.3, 4.4.3, 4.4.3, 4.4.3, 4.4.3, 4.4.3

[19] S. Chen, E. Dobriban, and J. H. Lee. A group-theoretic framework for data augmentation, 2019. 3.2.2

[20] C. Clark, M. Yatskar, and L. Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. *arXiv preprint arXiv:1909.03683*, 2019. 3.2.6

[21] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016. 3.2.6

[22] G. Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017. 3.1, 3.3.1, 4.4.1

[23] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. A1.3

[24] M. Cuturi and A. Doucet. Fast computation of wasserstein barycenters. 2014. 3.2.3

[25] T. Dao, A. Gu, A. Ratner, V. Smith, C. D. Sa, and C. Ré. A kernel theory of modern data augmentation. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1528–1537. PMLR, 2019. 3.2.2

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2.2.1

[27] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 2.2.1

[28] J. S. Denker, W. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon. Neural network recognizer for hand-written zip code digits. In *Advances in neural information processing systems*, pages 323–331, 1989. 3.3.2

[29] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 4.4.1, 4.4.3, 4.4.3

[30] S. Dhouib, I. Redko, and C. Lartizien. Margin-aware adversarial domain adaptation with optimal transport. In *Thirty-seventh International Conference on Machine Learning*, 2020. 4.1, 4.2.1

[31] Z. Ding and Y. Fu. Deep domain generalization with structured low-rank constraint. *IEEE Transactions on Image Processing*, 27(1):304–313, 2018. 3.3.1

[32] Q. Dou, D. C. Castro, K. Kamnitsas, and B. Glocker. Domain generalization via model-agnostic learning of semantic features. *arXiv preprint arXiv:1910.13580*, 2019. 4.4.3, 4.4.3

[33] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. 2.2.2

[34] S. Erfani, M. Baktashmotlagh, M. Moshtaghi, V. Nguyen, C. Leckie, J. Bailey, and R. Kotagiri. Robust domain generalisation by enforcing distribution invariance. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1455–1461. AAAI Press/International Joint Conferences on Artificial Intelligence, 2016. 3.3.1

[35] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 5.3

[36] A. Fawzi, H. Samulowitz, D. S. Turaga, and P. Frossard. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016*, pages 3688–3692. IEEE, 2016. 3.2.2, 4.3.1

[37] A. Galloway, A. Golubeva, T. Tanay, M. Moussa, and G. W. Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*, 2019. 2.2.2

[38] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014. 3.3.2

[39] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 3.2.2, 3.3.2, 3.3.2, 3.3.2, 3.3.3, 4.3.2

[40] X. Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 4.4.1

[41] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. 1.1, 3.2.1, 3.2.6, 3.4, 4.1, 4.2.2

[42] P. Germain, A. Habrard, F. Laviolette, and E. Morvant. A new pac-bayesian perspective on domain adaptation. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages

859–868. JMLR.org, 2016. 4.1, 4.2.1

[43] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018. 4.4.1, 4.4.3, 4.4.3

[44] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015. 3.3.1, 3.3.2, 3.3.2, 4.4.1

[45] A. Ghosh and A. H. Thiery. On data-augmentation and consistency-based semi-supervised learning. In *International Conference on Learning Representations*, 2021. 3.2.2

[46] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. PMLR, 2010. 2.2.1

[47] K. Goel, A. Gu, Y. Li, and C. Re. Model patching: Closing the subgroup performance gap with data augmentation. In *International Conference on Learning Representations*, 2021. 3.2.1

[48] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 3.2.2

[49] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples (2014). In *International Conference on Learning Representations*, 2015. 2.1, 2.2.3, 3.2.3, 4.2.2

[50] A. Gretton, A. J. Smola, J. Huang, M. Schmittfull, K. M. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. *Journal of Machine Learning Research*, 2009. 3.3.1

[51] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans, 2017. 3.2.2, 3.2.4

[52] C. Guo, J. S. Frank, and K. Q. Weinberger. Low frequency adversarial perturbation, 2018. 2.2.5

[53] H. Guo, K. Zheng, X. Fan, H. Yu, and S. Wang. Visual attention consistency under image transforms for multi-label image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 729–739. Computer Vision Foundation / IEEE, 2019. 3.2.2

[54] R. M. Haralick, K. Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, 1973. 3.3.2

[55] H. He, S. Zha, and H. Wang. Unlearn dataset bias in natural language inference by fitting the residual. *arXiv preprint arXiv:1908.10763*, 2019. 1.1

[56] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1.1

[57] 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. 2.2.1, 2.2.2, 2.2.4

[58] J. J. Heckman. Sample selection bias as a specification error (with an application to the estimation of labor supply functions), 1977. 3.3.1

[59] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019. 3.4, A1.3

[60] 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. In *International Conference on Learning Representations*, 2019. 1.1

[61] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019. 3.2.6

[62] 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. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 3.2.1, 3.2.2, A1.3

[63] K. Hermann, T. Chen, and S. Kornblith. The origins and prevalence of texture bias in convolutional neural networks. *Advances in Neural Information Processing Systems*, 33, 2020. 1.1, 2.3

[64] A. Hernández-García and P. König. Data augmentation instead of explicit regularization, 2018. 3.2.2

[65] G. E. Hinton, A. Krizhevsky, I. Sutskever, and N. Srivastva. System and method for addressing overfitting in a neural network, Aug. 2 2016. US Patent 9,406,017. 2.2.2

[66] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2731–2741. PMLR, 2019. 3.2.2

[67] W. Hu, G. Nio, I. Sato, and M. Sugiyama. Does distributionally robust supervised learning give robust classifiers? *arXiv preprint arXiv:1611.02041*, 2016. 3.3.1

[68] Z. Hu, B. Tan, R. R. Salakhutdinov, T. M. Mitchell, and E. P. Xing. Learning data manipulation for augmentation and weighting. In *Advances in Neural Information Processing Systems*, pages 15738–15749, 2019. 3.2.2

[69] Z. Huang, H. Wang, E. P. Xing, and D. Huang. Self-challenging improves cross-domain generalization. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pages 124–140. Springer, 2020. 3.2.2, A1.3

[70] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019. 2.3,

3.2.6

[71] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2.2.2

[72] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 3.2.6

[73] H. Jalalzai, P. Colombo, C. Clavel, É. Gaussier, G. Varni, E. Vignon, and A. Sabourin. Heavy-tailed representations, text polarity classification & data augmentation. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 3.2.2

[74] J. Jeong, S. Lee, J. Kim, and N. Kwak. Consistency-based semi-supervised learning for object detection. In *Advances in Neural Information Processing Systems*, pages 10758–10767, 2019. 3.2.2

[75] J. Jo and Y. Bengio. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017. 2.2.5, 3.3.2, 4.1, 4.2.2

[76] H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing, 2018. 3.2.2

[77] D. Kaushik, E. Hovy, and Z. C. Lipton. Learning the difference that makes a difference with counterfactually augmented data. In *International Conference on Learning Representations (ICLR)*, 2020. 3.2.1

[78] J. Kim, W. Choo, H. Jeong, and H. O. Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. In *International Conference on Learning Representations*, 2021. 3.2.2

[79] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2.2.1, 2.2.2, 3.3.2

[80] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017. 5.3

[81] R. Kondor, Z. Lin, and S. Trivedi. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 10117–10126, 2018. 3.2.6

[82] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 2.2.1

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1.1, 2.2.2

[84] A. Kumagai and T. Iwata. Zero-shot domain adaptation without domain semantic descriptors. *arXiv preprint arXiv:1807.02927*, 2018. 3.3.1

[85] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In

Workshop of International Conference on Learning Representations, 2017. 2.2.3

[86] S.-C. Lam. Texture feature extraction using gray level gradient based co-occurence matrices. In *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, volume 1, pages 267–271. IEEE, 1996. 3.3.2, 3.3.2

[87] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 4.4.1

[88] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2.2.2, 3.2.2, 3.3.2

[89] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1019–1030. Curran Associates, Inc., 2018. 3.3.3

[90] S. Lee, D. Kim, N. Kim, and S.-G. Jeong. Drop to adapt: Learning discriminative features for unsupervised domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 91–100, 2019. 4.4.1, 4.4.3

[91] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5542–5550, 2017. 3.3.2, 3.3.3, 3.4, 4.4.3, 4.4.3, 4.6

[92] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Learning to generalize: Meta-learning for domain generalization. *arXiv preprint arXiv:1710.03463*, 2017. 3.3.1, 3.3.2, 3.3.3

[93] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales. Episodic training for domain generalization. *arXiv preprint arXiv:1902.00113*, 2019. 4.4.1, 4.4.3, 4.4.3

[94] H. Li, S. J. Pan, S. Wang, and A. C. Kot. Domain generalization with adversarial feature learning. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, 2018. 3.3.1, 4.4.1

[95] W. Li, Z. Xu, D. Xu, D. Dai, and L. Van Gool. Domain generalization and adaptation using low rank exemplar svms. *IEEE transactions on pattern analysis and machine intelligence*, 2017. 3.3.1

[96] D. Liang, Z. Huang, and Z. C. Lipton. Learning noise-invariant representations for robust speech recognition. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 56–63. IEEE, 2018. 3.2.2

[97] P. Liang. Cs229t/stat231: Statistical learning theory (winter 2016), 2016. A1.2

[98] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2.2.4

[99] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2.2.4

[100] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection.

In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2.2.4

[101] Z. C. Lipton, Y.-X. Wang, and A. Smola. Detecting and correcting for label shift with black box predictors. In *International Conference on Machine Learning (ICML)*, 2018. 3.3.1

[102] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019. A1.3

[103] W.-S. Lu. Wavelet approaches to still image denoising. In *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No. 97CB36136)*, volume 2, pages 1705–1709. IEEE, 1997. 2.2.5

[104] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 1.1, 2.2.2, 3.2.2, 3.2.5, 4.3.1, 5.2, A1.1

[105] R. K. Mahabadi, Y. Belinkov, and J. Henderson. End-to-end bias mitigation by modelling biases in corpora. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8706–8716. Association for Computational Linguistics, 2020. 1.1

[106] M. Mancini, S. R. Bulò, B. Caputo, and E. Ricci. Best sources forward: domain generalization through source-specific nets. *arXiv preprint arXiv:1806.05810*, 2018. 3.3.1, 3.3.2, 3.3.3

[107] C. F. Manski and S. R. Lerman. The estimation of choice probabilities from choice based samples. *Econometrica: Journal of the Econometric Society*, 1977. 3.3.1

[108] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009. 3.3.1, 4.1, 4.2.1

[109] J. McDonald. Confounding variables. *Handbook of biological statistics, 3rd edn. Sparky House Publishing, Baltimore*, pages 24–28, 2014. 1.1, 2

[110] T. M. Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37): 870–877, 1997. 4.4.2

[111] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino. Curriculum dropout. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3544–3552, 2017. 4.4.2

[112] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2, page 3, 2017. 3.3.1, 3.3.2

[113] D. Moyer, S. Gao, R. Brekelmans, G. V. Steeg, and A. Galstyan. Evading the adversary in invariant representation. *arXiv preprint arXiv:1805.09458*, 2018. 3.3.2

[114] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18, 2013. 3.1, 3.2.3, 3.3.1, 4.4.1

[115] N. Nangia and S. R. Bowman. Human vs. muppet: A conservative estimate of human performance on the glue benchmark. *arXiv preprint arXiv:1905.10425*, 2019. 1.1

[116] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011. 3.3.2

[117] L. Niu, W. Li, and D. Xu. Multi-view domain generalization for visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4193–4201, 2015. 3.3.1

[118] S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992. 4.4.1

[119] S. Park and N. Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian conference on computer vision*, pages 189–204. Springer, 2016. 4.4.1, 4.4.3

[120] S. Park, J. Park, S.-J. Shin, and I.-C. Moon. Adversarial dropout for supervised and semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 4.4.1, 4.4.3, 4.4.3

[121] S. S. Platonov. The fourier transform of functions satisfying the lipschitz condition on rank 1 symmetric spaces. *Siberian Mathematical Journal*, 46(6):1108–1118, 2005. 2.2.3

[122] Y. Qu, D. Shen, Y. Shen, S. Sajeev, W. Chen, and J. Han. Co{da}: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. In *International Conference on Learning Representations*, 2021. 3.2.2

[123] QuickDraw. Quick draw! the data, 2018. URL `https://quickdraw.withgoogle.com/data`. 3.4

[124] N. Rahaman, D. Arpit, A. Baratin, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of deep neural networks. *arXiv preprint arXiv:1806.08734*, 2018. 2.2.1

[125] S. Rajput, Z. Feng, Z. Charles, P.-L. Loh, and D. Papailiopoulos. Does data augmentation lead to positive margin? In *International Conference on Machine Learning*, pages 5321–5330, 2019. 3.2.2

[126] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet?, 2019. 3.4

[127] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 2

[128] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011. 3.3.2

[129] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. 4.4.3, 4.4.4

[130] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 3.3.2

[131] 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. *arXiv preprint arXiv:1911.08731*, 2019. 1.1

[132] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*, pages 1163–1171, 2016. 3.2.2

[133] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016. 3.4

[134] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. On causal and anticausal learning. In *International Coference on International Conference on Machine Learning (ICML-12)*, pages 459–466. Omnipress, 2012. 3.3.1

[135] M. Shah, X. Chen, M. Rohrbach, and D. Parikh. Cycle-consistency for robust visual question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3.2.2

[136] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018. 3.3.2, 4.4.1

[137] Y. Sharma, G. W. Ding, and M. Brubaker. On the effectiveness of low frequency perturbations, 2019. 2.2.5

[138] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 2000. 3.3.1

[139] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019. 3.2.2

[140] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2.2.2

[141] K. K. Singh and Y. J. Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *2017 IEEE international conference on computer vision (ICCV)*, pages 3544–3553. IEEE, 2017. 4.4.1

[142] A. Sinha, K. Ayush, J. Song, B. Uzkent, H. Jin, and S. Ermon. Negative data augmentation. In *International Conference on Learning Representations*, 2021. 3.2.2

[143] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning*

research, 15(1):1929–1958, 2014. 4.4.1

[144] A. Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 2009. 3.3.1

[145] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1.1, 2.1, 3.2.1, 3.2.3

[146] K. S. Tai, P. Bailis, and G. Valiant. Equivariant transformer networks. *arXiv preprint arXiv:1901.11399*, 2019. 3.2.6

[147] K. Tian, C. Lin, M. Sun, L. Zhou, J. Yan, and W. Ouyang. Improving auto-augment via augmentation-wise weight sharing. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 3.2.2

[148] E. C. Titchmarsh. Introduction to the theory of fourier integrals. 1948. 2.2.3

[149] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015. 4.4.1

[150] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011. 1.1, 2, 4.4.3, 4.7

[151] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019. 3.2.4

[152] Z. Tu, J. Zhang, and D. Tao. Theoretical analysis of adversarial learning: A minimax approach. In *Advances in Neural Information Processing Systems*, pages 12280–12290, 2019. A1.1

[153] A. F. M. S. Uddin, M. S. Monira, W. Shin, T. Chung, and S.-H. Bae. Saliencymix: A saliency guided data augmentation strategy for better regularization. In *International Conference on Learning Representations*, 2021. 3.2.2

[154] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017. 4.4.3, 4.8

[155] T. Vigen. *Spurious correlations*. Hachette books, 2015. 1.1, 2

[156] C. Villani. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003. A1.2

[157] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. 3.2.3

[158] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, pages 5334–5344, 2018. 4.4.1

[159] H. Wang and B. Raj. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017. 2.2.1

[160] H. Wang, A. Meghawat, L.-P. Morency, and E. P. Xing. Select-additive learning: Improving generalization in multimodal sentiment analysis. *arXiv preprint arXiv:1609.05244*, 2016. 3.3.1

[161] H. Wang, Z. Wu, and E. P. Xing. Removing confounding factors associated weights in deep neural networks improves the prediction accuracy for healthcare applications. In *BIOCOMPUTING 2019: Proceedings of the Pacific Symposium*, pages 54–65. World Scientific, 2018. 1.1

[162] H. Wang, S. Ge, Z. C. Lipton, and E. P. Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 10506–10518, 2019. 1.1, 3.2.6, 4.3.2, 4.4.1, 4.4.3, 4.4.3, 4.4.3, 4.9, 4.4.3, A1.3

[163] H. Wang, Z. He, Z. C. Lipton, and E. P. Xing. Learning robust representations by projecting superficial statistics out. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 1.1, 2.2.5, 3.1, 3.3.3, 3.3.3, 3.3.3, 4.4.1, 4.4.3, 4.4.3, 4.4.3, A1.3

[164] H. Wang, T. Yue, J. Yang, W. Wu, and E. P. Xing. Deep mixed model for marginal epistasis detection and population stratification correction in genome-wide association studies. *BMC bioinformatics*, 20(23):1–11, 2019. 1.1

[165] H. Wang, Z. Huang, X. Wu, and E. P. Xing. Squared ℓ_2 norm as consistency loss for leveraging augmented data to learn robust and invariant representations. *arXiv preprint arXiv:2011.13052*, 2020. 1.1

[166] H. Wang, X. Wu, Z. Huang, and E. P. Xing. High-frequency component helps explain the generalization of convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8684–8694, 2020. 3.1, 3.2.1, 3.2.4, 3.2.4, 4.1, 4.2.2, 4.4.1

[167] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312: 135–153, 2018. 3.1, 4.4.1

[168] Y. Wang, X. Pan, S. Song, H. Zhang, G. Huang, and C. Wu. Implicit semantic data augmentation for deep networks. In *Advances in Neural Information Processing Systems*, pages 12614–12623, 2019. 3.2.2

[169] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016. 3.3.1

[170] X. Wu, Y. Mao, H. Wang, X. Zeng, X. Gao, E. P. Xing, and M. Xu. Regularized adversarial training (RAT) for robust cellular electron cryo tomograms classification. In I. Yoo, J. Bi, and X. Hu, editors, *2019 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2019, San Diego, CA, USA, November 18-21, 2019*, pages 1–6. IEEE, 2019. 3.2.2

[171] Y. Wu, E. Winston, D. Kaushik, and Z. Lipton. Domain adaptation with asymmetrically-relaxed distribution alignment. In *International Conference on Machine Learning*, pages 6872–6881, 2019. 4.1

[172] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 2.2.1

[173] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*, 2019. 3.2.2

[174] S. Xie, T. Yang, X. Wang, and Y. Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2645–2654, 2015. 3.2.2

[175] F. Yang, Z. Wang, and C. Heinze-Deml. Invariance-inducing regularization using worst-case transformations suffices to boost accuracy and spatial robustness. In *Advances in Neural Information Processing Systems*, pages 14757–14768, 2019. 3.2.2

[176] M. Yi, L. Hou, L. Shang, X. Jiang, Q. Liu, and Z.-M. Ma. Reweighting augmented samples by minimizing the maximal expected loss. In *International Conference on Learning Representations*, 2021. 3.2.2

[177] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 2.2.2

[178] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. 2.1, 2.2.1

[179] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2.2.2

[180] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 2019. 3.2.2, 3.2.4

[181] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, 2013. 3.3.1

[182] L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2021. 3.2.2

[183] R. Zhang. Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*, 2019. A1.3

[184] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong. Adversarial autoaugment. In *International Conference on Learning Representations*, 2020. 3.2.2

[185] Y. Zhang, T. Liu, M. Long, and M. I. Jordan. Bridging theory and algorithm for domain adaptation. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach,*

California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 7404–7413. PMLR, 2019. 4.1, 4.2.1

[186] Z. Zhang, S. Wu, S. Liu, M. Li, M. Zhou, and T. Xu. Regularizing neural machine translation by target-bidirectional agreement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 443–450, 2019. 3.2.2

[187] H. Zhao, R. T. des Combes, K. Zhang, and G. J. Gordon. On learning invariant representations for domain adaptation. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7523–7532. PMLR, 2019. 4.1

[188] L. Zhao, T. Liu, X. Peng, and D. N. Metaxas. Maximum-entropy adversarial data augmentation for improved generalization and robustness. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 3.2.2

[189] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016. 3.2.2

[190] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le. Learning data augmentation strategies for object detection, 2019. 3.2.2