Learning Unforeseen Robustness from Out-of-distribution Data Using Equivariant Domain Translator

Sicheng Zhu ¹ Bang An ¹ Furong Huang ¹ Sanghyun Hong ²

Abstract

Current approaches for training robust models are typically tailored to scenarios where data variations are accessible in the training set. While shown effective in achieving robustness to these foreseen variations, these approaches are ineffective in learning unforeseen robustness, i.e., robustness to data variations without known characterization or training examples reflecting them. In this work, we learn unforeseen robustness by harnessing the variations in the abundant out-of-distribution data. To overcome the main challenge of using such data, the domain gap, we use a domain translator to bridge it and bound the unforeseen robustness on the target distribution. As implied by our analysis, we propose a two-step algorithm that first trains an equivariant domain translator to map out-of-distribution data to the target distribution while preserving the considered variation, and then regularizes a model’s output consistency on the domain-translated data to improve its robustness. We empirically show the effectiveness of our approach in improving unforeseen and foreseen robustness compared to existing approaches. Additionally, we show that training the equivariant domain translator serves as an effective criterion for source data selection.

1. Introduction

A trustworthy machine learning system should provide consistent output despite nuisance transformations in input. For instance, a self-driving car should consistently recognize road objects, regardless of viewpoint changes that do not alter the object’s label. This desirable property of a model is measured by robustness — a hallmark feature exhibited by humans and numerous other creatures (Tacchetti et al., 2018). Training a model to be robust not only improves its trustworthiness but may also improve its in-distribution (Zhou et al., 2022) and out-of-distribution (OOD) generalization (Hendrycks et al., 2020), potentially by expanding the labeled region (Wei et al., 2021).

Recognizing the importance of robustness, previous work has proposed several methods to train robust models (a.k.a. robustness interventions), including training on augmented data (Sohn et al., 2020), consistency regularization (Xie et al., 2020), adversarial training (Madry et al., 2018), and architecture modifications (Zhang, 2019). These methods effectively improve robustness against foreseen data variations — those that can be characterized by known transformation functions or observable in pairs of training examples before and after the transformation, such as noise corruption (Hendrycks & Dietterich, 2019) and spatial transformations (Engstrom et al., 2019).

Nevertheless, unforeseen robustness remains challenging to achieve, with existing methods either unable or struggling to learn it. This issue is particularly problematic given that in many datasets, only specific synthetic data variations are foreseen, while others, including most natural variations, are not. As a result, models remain vulnerable to these unforeseen data variations, such as changes in viewpoint (Koh et al., 2021) or time (Shankar et al., 2021).

This work introduces a method to learn unforeseen robustness. Notably, the data variation unforeseen from a given training set is often observable as pairs of transformed examples in the abundant out-of-distribution data, such as simulations. Leveraging this observation, we propose to learn unforeseen robustness from out-of-distribution data, using an equivariant domain translator to bridge the domain gap while preserving the variation, as illustrated in Figure 1.

Contributions: First, we formulate the problem of learning unforeseen robustness from out-of-distribution data (§3.1) and identify the difficulties in extending existing approaches to solve this problem (Figure 1).

Second, recognizing that the primary challenge of this problem stems from the domain gap, we analyze the problem
Learning Unforeseen Robustness from Out-of-distribution Data Using Equivariant Domain Translator

2. Related Work

Semi-supervised consistency regularization. A large body of work uses consistency regularization for semi-supervised learning (Sohn et al. (2020)), achieving state-of-the-art results in generalization. The key idea is to do supervised learning on the labeled data while regularizing the model to predict consistently on the unlabeled data, which potentially expands the labeled region and thus improves generalization (Wei et al., 2021). Despite the various goals previous work has, such as improving generalization (Sohn et al., 2020) or improving adversarial robustness (Zhang et al., 2020), we focus on solving the problem of learning unforeseen robustness.

with an auxiliary domain translator bridging the gap (§3.2). By considering a domain translator, i.e., a map from the input space to itself, we establish an upper bound for the robustness loss on the target distribution in terms of variations on the source distribution. In particular, this bound can be tightened by a domain translator that has two properties: equivariant, meaning that transforming an example first and then domain-translating it yields similar output as domain-translating the example first and then transforming it, and accurate, meaning that the domain-translated source distribution closely aligns with the target distribution in terms of the Wasserstein-1 distance.

Third, we propose a two-step algorithm for solving the problem based on our prior analysis (§4). The first step trains an equivariant and accurate domain translator. To make it accurate, we train the translator under the supervision of a Lipschitz-regularized domain discriminator, following WGAN (Arjovsky et al., 2017). To make it equivariant, we offer three optional regularization losses and choose one depending on our knowledge of the transformation function and the transformation parameter associated with each pair of transformed examples. The second step uses consistency regularization on the domain-translated source data to improve a model’s robustness.

Fourth, we empirically evaluate our method for image classification tasks on a combination of seven source datasets, two target datasets, and two types of data variations (§5). We first verify that our method indeed learns equivariant and accurate domain translators. Then, we show the effectiveness of our method in learning unforeseen robustness compared to other baselines, and further support it by ablation studies. As a by-product, we also show that the training result of the equivariant domain translator correlates strongly (R=0.91) with the robustness benefit of a certain source dataset, indicating its usefulness as a source dataset selection criterion.

Fifth, we apply our method to two real-world tasks to demonstrate its practical significance. First, we learn the 3D viewpoint change robustness on CIFAR-10 by harnessing variations in video clips and show the improvement using surrogate transformations. Second, we show that our method can leverage out-of-distribution data to further improve the unforeseen robustness on the target, effectively serving as a generalized and improved unsupervised data augmentation method. Our method achieves better improvements in robustness, in-distribution generalization, and out-of-distribution generalization compared to the previous method (Xie et al., 2020). We will make our code publicly available at https://github.com/schzhu/unforeseen-robustness.
In this section, we first formulate the problem of learning unforeseen robustness by harnessing variations on source data, which has rarely been addressed before. To quantify the potential robustness achievable through learning from source data, we then establish an upper bound for the robustness loss on the target distribution in terms of variations on the source.

3.1. Problem: Robustness from Variations on Source

In this problem, we are given some target examples \( \{ x_i \} \) sampled from the target data distribution \( \mathbb{P} \) on the input space \( \mathcal{X} \). We consider \( \mathcal{X} \) to be \( \mathbb{R}^d \) since we focus on image data. In addition, we are given some source examples \( \{ u_i \} \) sampled from the source data distribution \( \mathbb{Q} \) on \( \mathcal{X} \). We do learning over a family of models \( \{ f : \mathcal{X} \rightarrow \mathbb{R}^k \} \) which map examples in \( \mathcal{X} \) to \( k \)-dimensional output vectors. For classification tasks, we consider the model’s logit output (before softmax) as the model output.

Data variation. We consider data variations that can be represented by some (possibly unknown) data transformation function \( \phi : \mathcal{T} \times \mathcal{X} \rightarrow \mathcal{X} \), where \( \mathcal{T} \) is the set of possible transformation parameters. Some examples are noise corruption, group actions with \( \mathcal{T} \) being a group (e.g. flipping), and 3D viewpoint changes projected to the 2D pixel space (given that \( \phi \) models the stochasticity). As we focus on random data transformation, we also consider some transformation parameter distribution \( \mathbb{T} \) on \( \mathcal{T} \). We assume that the data variation is unforeseen, meaning that we neither know the data transformation function nor have transformed target example pairs \( \{(x_i, \phi_{t_i}(x_i))\} \). Instead, given the source examples \( \{ u_i \} \), we have finite (e.g., variations extracted from video clips) or infinite (e.g., simulated data) transformed versions \( \{ \phi_{t_i}(u_i) \} \), where \( t_{ij} \) is sampled from \( \mathbb{T} \).

Robustness. We consider model robustness to random data transformations. To measure the consistency of two model outputs, we use some loss function \( \ell : \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}_{\geq 0} \) that satisfies the triangle inequality \( \ell(v, v') \leq \ell(v, v'') + \ell(v'', v'), \forall v \in \mathbb{R}^k \). Examples of such loss functions include the zero-one loss \( \ell_0(v, v') = 1 \{ \max_i v_i \neq \max_i v_i' \} \), the \( \ell_p \) loss \( \ell_p(v, v') = \|v - v'\|_p \) for some \( p \geq 1 \), and certain f-divergences like the square root of JS-divergence (Endres & Schindelin, 2003). Given such a loss function, we define the following robustness loss.

Definition 3.1 (Robustness loss). Let \( \phi \) be some transformation function and \( \mathbb{T} \) be the distribution of transformation parameters. Then the robustness loss of a model \( f \) on the data distribution \( \mathbb{P} \) is defined as

\[
L_\phi(f, \mathbb{P}) = \mathbb{E}_{x \sim \mathbb{P}} \max_{t \sim \mathbb{T}} \left[ \ell(f(x), f(\phi_{t_i}(x))) \right]
\] (3.1)

Note that the robustness loss is label-agnostic, making it well-defined on domains with different label sets. Similar notions of robustness also appear in the literature (e.g., Hendrycks & Dietterich (2019) and Zhou et al. (2022)).
Goal. Given the target examples \(\{x_i\}\), and the source examples \(\{u_i\}\) with their transformed versions \(\{\phi_t(u_i)\}\), our goal is to learn a model \(f\) that minimizes the robustness loss on the target distribution \(L_\phi(f, P)\) in addition to some other primary task loss (e.g., classification loss). We refer to this problem as learning robustness on the target data from variations on the source data.

For classification tasks, the significance of minimizing the robustness loss is that a small robustness loss along with a small classification loss \(E_{x \sim P, t \sim T}[\ell_0(1, y, f(x))]\) are sufficient to guarantee a small robust classification loss \(E_{x \sim P, t \sim T}[\ell_0(1, y, f(\phi_t(x)))]\), where \(y\) is the ground-truth label of \(x\) (Zhang et al., 2019).

### 3.2. Robustness Guarantee with Domain Translator

Directly minimizing the robustness loss on the target distribution \(L_\phi(f, P)\) requires transformed target examples pairs \(\{x_i, \phi_t(x_i)\}\) to estimate the expectation. However, we lack these pairs and are unaware of the transformation function needed to sample them. In this case, the following proposition shows the feasibility of leveraging the available source examples with the help of a domain translator.

To simplify notation, we use \(\overline{\ell}_t : X \to \mathbb{R}\) to denote the function \(\overline{\ell}_t(x) := E_{x \sim T}[\ell(f(x), f(\phi_t(x)))]\), which intuitively measures the robustness loss of the model at a given example. Given some (measurable) function \(\xi : X \to X\), we use \(\xi#Q\) to denote the push-forward probability distribution\(^1\) of \(Q\) on \(X\). We use \(W_1\) to denote Wasserstein-1 distance.

**Proposition 3.2.** We assume that \(\overline{\ell}_t\) is Lipschitz uniformly over all models \(f\), with a (possibly infinite) Lipschitz constant \(\|\ell\|_L\). Then for any (measurable) function \(\xi : X \to X\), the following holds:

\[
L_\phi(f, P) \leq I_1 + I_2 + I_3,
\]

where

\[
I_1 = \mathbb{E}_{u \sim Q, t \sim T}[\ell(f(\xi(u)), f(\xi \circ \phi_t(u)))],
\]

\[
I_2 = \mathbb{E}_{u \sim Q, t \sim T}[\ell(f(\xi \circ \phi_t(u)), f(\phi_t \circ \xi(u)))]
\]

\[
I_3 = \|\ell\|_L W_1(P, \xi#Q).
\]

This proposition, proved in Appendix A.1, upper-bounds the robustness loss on the target distribution by three terms illustrated in Figure 2. We can intuitively interpret \(\xi\) as a domain translator which translates a given source example into another example that “looks like” the target examples\(^2\). Below, we remark on two properties of the domain translator.

**Equivariant domain translator minimizes \(I_2\).** Note that any domain translator \(\xi\) satisfying \(\xi \circ \phi_t(u) = \phi_t \circ \xi(u)\)

---

\(^1\)We state the definition in Appendix A.1.

\(^2\)Compared to the style transfer work, such translation is unpaired and does not need to preserve the underlying concept class.
where the first term minimizes $I_3$, encouraging accurate domain translation, and the second term minimizes $I_2$, encouraging equivariance. The hyperparameter $\lambda$ balances the two objectives.

In our implementation, we adopt the encoder-decoder architecture commonly used in style transfer literature as our domain translator $\xi$. To optimize the first term, we follow WGAN (Arjovsky et al., 2017) and train the domain translator $\xi$ under the supervision of an auxiliary domain discriminator that has regularized Lipschitz constant. To estimate and optimize the second term, we proceed as follows: we randomly sample one transformation parameter for each source example, apply domain translation followed by transformation to get $\phi_t \circ \xi(u)$, and apply transformation followed by domain translation to get $\xi \circ \phi_t(u)$. We encourage the domain translator to generate examples such that the two terms are similar in terms of the $\ell_2$ loss.

In addition to WGAN, some recent work suggests that score-based generative models also implicitly minimize the Wasserstein distance (Kwon et al., 2022). Nevertheless, we defer further exploration of alternative implementations, such as image-to-image diffusion models (Tumanyan et al., 2022; Bansal et al., 2022), to future work.

## 4.2. Methods for Encouraging Equivariance

Directly encouraging the equivariance (the second term in Eq. 4.1) for the domain translator $\xi$ requires knowing the data transformation function $\phi$. However, when learning unforeseen robustness, we only have some transformed source example pairs $\{(u_i, \phi_t(u_i))\}$ but lack knowledge about the underlying transformation function. This poses a challenge to encouraging equivariance since we cannot transform a domain-translated example $\xi(u)$ to get $\phi_t \circ \xi(u)$ in Eq. 4.1. To address this issue, we provide three optional methods for encouraging equivariance based on different assumptions about the available knowledge.

First, in some special cases where we know the transformation function, such as when using our algorithm to do unsupervised data augmentation, we can directly encourage equivariance by minimizing the equivariance loss in Eq. 4.1.

Second, if we do not know the transformation function but know the transformation parameters $\{t_i\}$ used to generate the pairs of transformed examples $\{(u_i, \phi_t(u_i))\}$, such as when learning unforeseen robustness from some simulated data, we can empirically encourage equivariance to the transformation $\phi_t$ by training a predictor that predicts the transformation parameters $\{t_i\}$ based on the model’s output. This method is proposed by some recent work (Qi et al., 2019; Lee et al., 2021; Dangovski et al., 2022).

Third, in cases where both the transformation function and transformation parameters are unknown, we propose an alternative method to encourage equivariance. The main idea is to encourage a learnable feature extractor to extract the same encoded information about the transformation parameter $\{t_i\}$ from both the original source example pairs $\{(u_i, \phi_t(u_i))\}$ and the domain translated source example pairs $\{(\xi(u_i), \xi \circ \phi_t(u_i))\}$. The intuition becomes more evident when we consider fixing the feature extractor using some hard-coded or pretrained model, such as an optical flow estimator. In such cases, we encourage the extraction of the same encoded transformation parameter from the two pairs, similar to the second method of predicting the transformation parameter. A more detailed description of this method appears in Appendix B.1.

## 4.3. Step Two: Training Robust Model

Our goal in this problem is to improve the robustness of a model while performing some primary task. To illustrate this, we consider the classification task with a given classification loss $L_{\text{classifier}}$. Building upon the prior training of the domain translator, which minimizes $I_2$ and $I_3$ in Proposition 3.2, we proceed to train a robust classifier $f$ to minimize $I_1$ and $L_{\text{classifier}}$ while freezing the translator.

For notation simplicity, we write $I_1$ as a functional of $f$ and $\xi$. We use $\xi_{id}$ to denote the trained domain translator, and use $\xi_{id}$ to denote the identity domain translator, which maps any example to itself (perfectly equivariant but not accurate). Then, the training objective is

$$\min_f L_{\text{classifier}}(f) + \lambda_1 I_1(f, \xi^*) + \lambda_2 I_3(f, \xi_{id}),$$

where $\lambda_1$ and $\lambda_2$ are weight hyperparameters. We include...
the last term in the objective, which is essentially consistency regularization on the source data, since we observe that this often produces the best result. In fact, the UDA method (Xie et al., 2020) can be viewed as a special case of our method, with $\lambda_1 = 0$ and $\lambda_2 = 1$.

5. Empirical Evaluation

In this section, we empirically verify the effectiveness of our two-step algorithm in learning unforeseen robustness. Different from Section 6, this section only considers synthetic data variations since they enable reliable robustness evaluation on the target dataset. To begin, we show that our equivariance-encouraging method effectively trains equivariant domain translators. Then, we compare the two-step algorithm with two existing methods extended to our setting and provide an ablation study. Lastly, we show that the training of the equivariant domain translator, as a byproduct, also serves as a criterion for selecting suitable source datasets for learning robustness. Our experimental settings are as follows:

Datasets. We use CIFAR-10 and CIFAR-100 as our target datasets, while selecting the source dataset from a range of options including SVHN, STL-10, CIFAR-100, MNIST, CelebA, and Caltech-256. Note that some source datasets, such as MNIST or CelebA, are visually distinct from the target datasets, mirroring real-world scenarios where the considered data variation is only available from extremely out-of-distribution data.

Data variations. We use two types of synthetic data variations: (1) RandAugment (Cubuk et al., 2020), which includes a diverse range of 14 random transformations, spanning from geometric transformations to color space changes, and their random combinations. The variety of these transformations allows us to evaluate our algorithm’s ability to preserve such variations; (2) Random rotation, as a supplementary evaluation due to its well-defined nature. Despite its simplicity, modeling random rotation using model-based methods has proven to be a challenging task (Zhou et al., 2022). To simulate the scenario of learning unforeseen robustness, we refrain from accessing the transformation function or transformed target example pairs during training. Instead, we only use the transformation function during testing to evaluate the learned robustness.

Other settings. To implement the domain translator, we adopt the encoder-decoder architecture borrowed from CycleGAN (Zhu et al., 2017), which comprises two down-sampling convolutional layers, two residual blocks for latent propagation, and two up-sampling convolutional layers. We use ResNet18 (He et al., 2016) to implement the classifier. We use cross-entropy loss for classification, KL-divergence for consistency regularizing, and mean-squared-error (MSE) loss for measuring the equivariance of the domain translator (the second term in Eq. 4.1). Unless otherwise stated, we set $\lambda = 1$ in Eq. 4.1 and $\lambda_1 = \lambda_2 = 0.5$ in Eq. 4.2. In this section, we do not apply data augmentation on the source or target to avoid entanglement with the considered data variation. We defer more setup details to Appendix C, and most results on random rotation and CIFAR-100 to Appendix D.

5.1. Training Equivariant Domain Translator

We first test if the equivariance-encouraging methods provided in Section 4.2 can train equivariant domain translators. Specifically, we compare three methods: (1) The baseline method, denoted as Std (standard), which does not apply any equivariance regularization by setting $\lambda = 0$ in Eq. 4.1; (2) The first optional method, denoted as EqGt (equivariant-ground-truth), which encourages equivariance using the ground-truth data transformation function; (3) The third optional method, denoted as Eq (equivariant), which does not require access to the transformation function or
We evaluate the classifiers using three metrics: (1) Robust accuracy (R), which measures the probability of a model preserving its prediction under input variations; (2) Robust Classification accuracy (RC), which measures the probability of a model predicting the correct label under input variations; (3) Standard accuracy (S), which measures the probability of a model predicting the correct label.

Our algorithm learns unforeseen robustness. Despite the stark dissimilarity between SVHN and CIFAR-10, Table 1 shows that both our algorithm and UDA can harness the variations on SVHN to improve the robust classification accuracy on CIFAR-10 by 4.1% and 3.2%, respectively, indicating the feasibility of learning unforeseen robustness from out-of-distribution data. Moreover, our algorithm improves the standard accuracy whereas UDA sometimes hurts it. Using SVHN, our algorithm increases the standard accuracy by 0.9% over ERM, whereas UDA falls short by 0.8%. Meanwhile, MBRDL underperforms ERM in all three metrics, indicating the importance of the learning methods. Our analysis of MBRDL in Appendix D.5 suggests that it may fail due to the domain gap and the difficulty in modeling complex variations in RandAugment.

Our algorithm consistently outperforms UDA. Since the consistency regularization in UDA and our algorithm intro-
Table 2. Ablation study of the two-step algorithm, varying whether to use the source dataset (Src) and the training method of the domain translator (DT). Using the equivariant domain translator plays a key role in learning unforeseen robustness.

<table>
<thead>
<tr>
<th>Src</th>
<th>DT</th>
<th>SVHN</th>
<th>STL-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RC (%)</td>
<td>S (%)</td>
</tr>
<tr>
<td>✓</td>
<td>EqGt</td>
<td>83.7 (↑ 0.5)</td>
<td>89.5</td>
</tr>
<tr>
<td>✓</td>
<td>Eq</td>
<td>83.2</td>
<td>89.9</td>
</tr>
<tr>
<td>✓</td>
<td>Std</td>
<td>82.8 (↑ 0.4)</td>
<td>88.5</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>82.3 (↑ 0.9)</td>
<td>88.2</td>
</tr>
<tr>
<td>×</td>
<td></td>
<td>79.1 (↓ 4.1)</td>
<td>89.0</td>
</tr>
</tbody>
</table>

duces an extra weight hyperparameter, we further vary that weight for a comprehensive comparison and show results in Figure 5. For both methods, we observe two stages as the regularization weight increases. In the first stage, increasing the weight improves both robust and standard accuracy. In the second, however, increasing the weight improves robust accuracy but hurts standard accuracy, leading to a robustness-accuracy trade-off. Nevertheless, our method outperforms UDA across all weight settings and achieves better Pareto-optimal in the second stage.

**Equivariant domain translator is the key.** Table 2 shows the ablation study result for the two-step algorithm. Compared to ERM (last row), harnessing variations from the source dataset (top four rows) improves the target robustness significantly. Furthermore, both EqGt and Eq outperform Std and the one not using the domain translator (fourth row), indicating the importance of using an equivariant domain translator. Among the top two rows, Eq shows comparable robust classification and standard accuracy to EqGt, indicating that our equivariance-encouraging method trains equivariant domain translators that are equally helpful for downstream classification.

5.3. Source Dataset Selection

When learning unforeseen robustness, we cannot use cross-validation to select suitable source datasets to learn from due to the lack of target data variations. For example, it is hard to determine whether using SVHN or CelebA would better improve the robustness to RandAugment on CIFAR-10. In this case, we show that the training result of an equivariant domain translator can serve as a selection criterion. Specifically, given the target dataset, the variation, and a source dataset, we evaluate three available selection criteria: (1) DT-Eq-FID, which trains an Eq domain translator and then computes the FID between the target dataset and the domain-translated source dataset. (2) DT-Std-FID, similar to DT-Eq-FID, but uses an Std-trained domain translator. (3) Naïve-FID, which directly computes the FID between the target and the source datasets. For all three criteria, we select source datasets with lower FIDs.

Figure 6. Correlation results for three source dataset selection criteria, with our method (first row) or UDA (second row) training the classifier on the corresponding source dataset. Each point represents a source dataset, where the x-coordinate is the score given by the criterion and the y-coordinate is the actual robust classification accuracy of the resulting classifier. We measure the Pearson correlation (R) and the p-value (p). DT-Eq-FID, which is based on our equivariant domain translator training, shows the strongest correlation among the three even when for UDA.

**Equivariant DT can select suitable source.** In Figure 6, we evaluate the three criteria for selecting source datasets for our method and UDA, DT-Eq-FID, based on our equivariant domain translator training, shows the strongest correlation among the three (R=-0.91 for our method, R=-0.94 for UDA) with the resulting classifier’s robustness, indicating its effectiveness as a general source dataset selection criterion. It favors CelebA over SVHN for learning robustness to RandAugment on CIFAR-10, which corroborates our results, whereas the other two criteria do not. DT-Eq-FID also has two desired properties compared to DT-Std-FID and Naïve-FID. It is sensitive to the considered data variation, while DT-Std-FID and Naïve-FID are not, enabling it to explain why the same source dataset can have different benefits for different data variations. It also depends on the order of the source and target datasets, enabling it to explain why SVHN as the source and CIFAR-10 as the target gives a worse result than the other way around.

6. Applications

Now, we apply our algorithm to two real-world tasks to show its practical significance. First, we train robust CIFAR-10 classifiers to unforeseen 3D-viewpoint changes. Then, we harness out-of-distribution data to further improve unforeseen robustness on the target data, resulting in improved in-distribution and out-of-distribution generalization.

6.1. Learning Unforeseen Robustness in Real-world

To evaluate the effectiveness of learning unforeseen real-world robustness, we choose Objectron (Ahmadyan et al.,
Table 3. Robust classification accuracy on CIFAR10 under six geometric data transformations, which serves as a surrogate for the 3D-viewpoint-change robustness. Our method best learns the unforeseen robustness to this natural variation.

<table>
<thead>
<tr>
<th>Variations</th>
<th>ERM (%)</th>
<th>UDA (%)</th>
<th>Ours (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affine</td>
<td>69.2 ± 0.5</td>
<td>69.7 (↑ 0.5)</td>
<td>70.9 (↑ 1.7)</td>
</tr>
<tr>
<td>Rotate</td>
<td>83.3 ± 0.3</td>
<td>83.4 (↑ 0.1)</td>
<td>84.5 (↑ 1.2)</td>
</tr>
<tr>
<td>Perspective</td>
<td>61.6 ± 0.6</td>
<td>54.8 (↓ 6.8)</td>
<td>63.2 (↑ 1.6)</td>
</tr>
<tr>
<td>Crop</td>
<td>85.5 ± 0.1</td>
<td>85.4 (↓ 0.1)</td>
<td>86.2 (↑ 0.7)</td>
</tr>
<tr>
<td>Elastic transform</td>
<td>85.9 ± 0.3</td>
<td>86.4 (↑ 0.5)</td>
<td>87.3 (↑ 1.4)</td>
</tr>
<tr>
<td>Fisheye</td>
<td>43.7 ± 1.2</td>
<td>33.9 (↓ 9.8)</td>
<td>43.8 (↑ 0.1)</td>
</tr>
<tr>
<td>Plate Spline</td>
<td>81.6 ± 0.3</td>
<td>81.4 (↓ 0.2)</td>
<td>82.8 (↑ 1.2)</td>
</tr>
</tbody>
</table>

Figure 7. For unforeseen variations, using our method in addition to data augmentation (DA) further improves robustness (RC and R), ID generalization (S), and OOD generalization (S on three OOD test sets). Compared to UDA, our method not only better improves robustness and in-distribution generalization, but also benefits OOD generalization while UDA cannot, demonstrating its superiority as an unsupervised data augmentation method.

Figure 7 shows our results. We further improve the robustness and in-domain generalization, doubling the improvement brought by UDA in the same setting. In addition, we improve the accuracy on three out-of-distribution datasets by 1.5%, 1.2%, and 2.4%, respectively, whereas UDA barely helps. This result demonstrates our method’s superiority for unsupervised data augmentation using unforeseen variations.

7. Conclusion

This paper introduces a new approach to learning robustness that broadens the scope of existing robustness interventions. Unlike previous methods confined to a limited range of data variations, our approach harnesses the variations observed in some source data to learn the robustness on the target data, thus expanding the spectrum of robustness types that can be effectively learned.

Limitations. The most evident limitation of our approach is the additional computational burden introduced by training the domain translator. Appendix B.2 provides an analysis of the computational cost compared to existing methods. This is mainly due to the need for training different domain translators for different source data. Therefore, we hope that future work can develop a foundation model for image-to-image translation, allowing our method to achieve almost cost-free domain translation by simply fine-tuning with the addition of equivariance regularization.

Another drawback of our approach is the need to find suitable source data for training, which is not always readily available as existing datasets are not constructed with our specific problem in mind. However, many datasets, particularly some simulation datasets (e.g., for autonomous driving), have the capability to exhibit real-world variations. Hence, we encourage the community to consider incorporating validation data with various variations when constructing datasets, both for learning and evaluating robustness.

Acknowledgments

Zhu, An, and Huang are supported by National Science Foundation NSF-IIS-FAI program, DOD-ONR-Office of Naval Research, DOD Air Force Office of Scientific Research, DOD-DARPA-Defense Advanced Research Projects Agency Guaranteeing AI Robustness against Deception (GARD), Adobe, Capital One and JP Morgan faculty fellowships.
References


Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. Exploring the landscape of spatial robustness. In ICML, 2019. 1


Griffin, G., Holub, A., and Perona, P. Caltech 256. 15


Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. CS 231N, 7(7):3, 2015. 15


Lu, S., Nott, B., Olson, A., Todeschini, A., Vahabi, H., Carmon, Y., and Schmidt, L. Harder or different? a closer look at distribution shift in dataset reproduction. In ICML Workshop on Uncertainty and Robustness in Deep Learning, 2020. 9, 15


11


Zhang, R. Making convolutional networks shift-invariant again. In ICML, 2019. 1


Appendix

Table of Contents

A Additional Analysis 13
  A.1 Proof of Proposition 3.2 ........................................... 13
  A.2 Discussion about the Existence of Equivariant and Accurate Domain Translators ................................................... 14

B Algorithm Details 14
  B.1 Detailed Method to Encourage Equivariance for Domain Translator ................................................................. 14
  B.2 Computational Complexity Analysis ........................................... 15
  B.3 Algorithm Pseudocode ........................................... 15

C Detailed Experimental Setup 15
  C.1 Our Method ................................................... 16
  C.2 MBRDL ................................................... 17
  C.3 UDA ................................................... 18

D Additional Results 18
  D.1 Visualizing the Results of Equivariant Domain Translator ................................................................. 18
  D.2 Results on CIFAR-100 ................................................... 18
  D.3 Additional Baselines ................................................... 18
  D.4 Sensitivities to Source Sample Size ................................................... 19
  D.5 Problems of MBRDL ................................................... 19

A. Additional Analysis

A.1 Proof of Proposition 3.2

Before giving the proof, we first state the definition of push-forward distribution, which appears in many textbooks (see, e.g., Koralov & Sinai (2007)).

**Definition A.1 (Push-forward distribution).** Given a probability space \((\Omega, \mathcal{F}, P)\), a measurable space \((\tilde{\Omega}, \tilde{\mathcal{F}})\), and a measurable mapping \(\xi: \Omega \to \tilde{\Omega}\), the push-forward distribution of \(P\) on the \(\sigma\)-algebra \(\tilde{\mathcal{F}}\) is defined by

\[
x_\# P(A) = P(\xi^{-1}(A)) \quad \text{for } A \in \tilde{\mathcal{F}},
\]

where \(\xi^{-1}(A) = \{\omega \in \Omega : \xi(\omega) \in A\}\) denotes the pre-image of a measurable set \(A\).

The proof follows from the assumptions that the loss \(\ell\) satisfies the triangle inequality and \(\bar{\ell}_f\) is Lipschitz uniformly over all models \(f\). Since we are working on \((\mathbb{R}^d, B(\mathbb{R}^d))\) with functions implemented by neural networks (with continuous activation functions) and common losses, we omit the measurability issue.

**Proof.** First, since \(\ell\) is non-negative, by Tonelli’s theorem, we have

\[
L_\phi(f, P) := \mathbb{E}_{x \sim P, \phi \sim T} \left[ \ell(f(x), f(\phi(x))) \right] = \mathbb{E}_{x \sim P} \left[ \bar{\ell}_f(x) \right],
\]

where \(\bar{\ell}_f(x) := \mathbb{E}_{t \sim T}[\ell(f(x), f(\phi_t(x)))].\)
Then, since $\bar{\ell}_f$ is uniformly Lipschitz with a Lipschitz constant $\|\bar{\ell}_f\|_L$, by Kantorovich-Rubenstein duality theorem (see, e.g., (Villani, 2021)), we have

$$\mathbb{E}_{x \sim P} [\bar{\ell}_f(x)] - \mathbb{E}_{x \sim Q} [\bar{\ell}_f(x)] \leq \|\bar{\ell}_f\|_W \mathbb{W}_1(P, Q).$$

Thirdly, since $\xi \# Q$ is the push-forward distribution of $Q$ through the mapping $\xi$, by change of measure, we have

$$\mathbb{E}_{x \sim \xi \# Q} [\bar{\ell}_f(x)] = \mathbb{E}_{u \sim Q} [\bar{\ell}_f(\xi(u))].$$

Lastly, since $\ell$ satisfies the triangle inequality, we have

$$\mathbb{E}_{u \sim Q} [\ell(\xi(u))] = \mathbb{E}_{u \sim Q} \mathbb{E}_{t \sim T} \left[ f(\xi(u)), f(\phi_t \circ \xi(u)) \right] \leq \mathbb{E}_{u \sim Q} \mathbb{E}_{t \sim T} \left[ f(\xi(u)), f(\phi_t \circ u) \right] + \mathbb{E}_{u \sim Q} \mathbb{E}_{t \sim T} \left[ f(\phi_t \circ \xi(u)), f(\phi_t \circ \xi(u)) \right].$$

Rearranging terms completes the proof.

A.2. Discussion about the Existence of Equivariant and Accurate Domain Translators

We discuss some of our conjectures about the existence here and leave the complete characterization to future work. Since we use continuous maps to instantiate $\xi$, we conjecture that the equivariant domain translator does not exist if the support of the source data distribution, after being expanded by the transformation, has a smaller intrinsic dimension (see, e.g., Pope et al. (2021); Salmona et al. (2022)) than that of the target. Indeed, we empirically observe that for some source and target datasets such as SVHN to CIFAR-10, training the domain translator yields a trade-off between the equivariance and the approximate performance, but such trade-off mitigates if we swap the source and target datasets. Interestingly, this existence issue seems to enable us to use the training result of an equivariant domain translator as the source selection criterion.

B. Algorithm Details

B.1. Detailed Method to Encourage Equivariance for Domain Translator

Figure 8 illustrates our proposed method and discusses the intuition behind it. Compared to previous work, our method only requires the transformed source example pairs and their domain-translated counterparts. To use it in training the domain translator, we replace the second term in Eq. 4.1 with the cosine similarity loss shown in the figure. We simultaneously train the domain translator, projector, and predictor, to minimize the loss. This method applies to any data transformation that can be represented as $\phi_t$, without needing to modify the architecture or hyperparameters. When we know the transformation’s type (e.g., motion changes across video frames), we may also hard-code the predictor accordingly (e.g., use an optical-flow estimator) for better performance.

Figure 8. Illustration of our proposed method for encouraging the equivariance of the domain translator, requiring neither the transformation function $\phi_t$ nor its parameter $t$. Here, the projector, whose architecture refers to Qi et al. (2019), takes as input the original example $u$ and its transformed version $\phi_t(u)$ and outputs a vector $z_1$. The intuition is that $z_1$ may contain the encoded transformation parameter, which is exactly the case when the projector is a hard-coded model like an optical flow estimator. If the domain translator is equivariant, then the domain-translated pair $\xi(u)$ and $\xi(\phi_t(u))$ should also contain the same encoded transformation parameter. Thus, we encourage $z_1$ and $z_2$ to be similar, which is implemented with a predictor to prevent degeneration (referring to SimSiam (Chen & He, 2021)).
B.2. Computational Complexity Analysis

We report computational complexity in Table 4 and 5. When training the classifier, our method requires approximately 24% more time compared to UDA and MBRDL. When only using the trained domain translator, the required time is similar to that of UDA and MBRDL. We note that our method can potentially be accelerated by pre-translating all source examples and implementing proper parallelization techniques.

<table>
<thead>
<tr>
<th>Complexity</th>
<th>GPU seconds per epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>encoder-decoder</td>
<td>discriminator</td>
</tr>
<tr>
<td>UDA</td>
<td>n/a</td>
</tr>
<tr>
<td>MBRDL</td>
<td>4, 2</td>
</tr>
<tr>
<td>Ours</td>
<td>3, 1</td>
</tr>
<tr>
<td>WGAN (with encoder-decoder)</td>
<td>2, 1</td>
</tr>
</tbody>
</table>

B.3. Algorithm Pseudocode

We show the pseudocode of training the domain translator and classifier in Algorithm 1 and 2.

C. Detailed Experimental Setup

Datasets. In Section 5, we use CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) as target datasets and SVHN (Netzer et al., 2011), STL-10 (Coates et al., 2011), CIFAR-100, MNIST (Deng, 2012), CelebA (Liu et al., 2015), and Caltech-256 (Griffin et al.). When training domain translators, we only use unlabeled images from the source and target. In Section 6, we use Objectron (Ahmadyan et al., 2021) as the source dataset to learn 3D-viewpoint-change robustness. Objectron is a collection of short, object-centric video clips. We randomly sample several frames from each clip as the anchor images and randomly sample frames in a range of 10 frames as the 3D-viewpoint changed images. We use such pairs to do 3D-viewpoint change consistency regularization. To evaluate the out-of-distribution generalization of classifiers trained on CIFAR-10, we use CIFAR-10.1 (Recht et al., 2018), CIFAR-10.2 (Lu et al., 2020), and CIFAR-10-C (Hendrycks & Dietterich, 2019) as the ood datasets. CIFAR-10.1 and CIFAR-10.2 are sampled from TinyImageNet (Le & Yang, 2015) with the same classes of CIFAR-10. CIFAR-10-C is a collection of a corrupted version of CIFAR-10 under 15 types of corruption.

Data variations. In Section 5, we use RandAugment and random rotation as the variations. RandAugment contains 14 candidate transformation functions: “ShearX”, “ShearY”, “TranslateX”, “TranslateY”, “Rotate”, “Brightness”, “Color”, “Contrast”, “Sharpness”, “Posterize”, “Solarize”, “AutoContrast”, “Equalize”, and “Identity”. When using RandAugment, a composition of two randomly selected functions are applied to the images. For random rotation, we use $[-30^\circ, 30^\circ]$ random rotation. Although the rotation is simply defined, it cannot be modeled by existing model-based methods that use MUNIT-like architectures (Zhou et al., 2022). In Section 6, we consider 3D-viewpoint change as the unforeseen variation. We randomly select two nearby frames from one video clip as the two 3D-views of one object. Since we could not evaluate the model robustness to 3D-viewpoint change on the target data (CIFAR-10), we use six proxy transformations to estimate the 3D-viewpoint robustness. Proxy transformations are geometric transformations that do warping on images, which include “Random Affine”, “Random Rotate”, “Random Perspective”, “Random Crop”, “Random Fisheye”, “Random Thin Plate Spline”.

Algorithm 1 Training the domain translator (PyTorch-style pseudocode)

**Input:** domain translator $\xi$, discriminator $\text{disc}$, projector $\text{proj}$, and predictor $\text{pred}$
- target data $\{x_i\}_{i=1}^N$, source data $\{(u_i, u'_i)\}_{i=1}^M$ (consists of transformed pairs),
- batch size $B$, coefficient $\lambda$ for equivariance regularization

**Output:** trained domain translator $\xi^*$

randomly initialize all modules

for epoch in range(max_training_epochs) do
  for a target batch $X_B = \{x_i\}_{i=1}^B$ in all target data do
    randomly sample a source batch $\{(u_i, u'_i)\}_{i=1}^B$
    ( denote $U_B = \{u_i\}_{i=1}^B$, $U'_B = \{u'_i\}_{i=1}^B$, $UU'_B = \{(u_i, u'_i)\}_{i=1}^B$, $\xi(U_B) = \{\xi(u_i)\}_{i=1}^B$ )

    # training discriminator $\text{disc}$
    translate the source batch to get $\xi(U_B)$
    $\text{loss}_{\text{disc}} = \text{disc}(\xi(U_B)).\text{mean}() - \text{disc}(X_B).\text{mean}()$
    update $\text{disc}$ according to $\text{loss}_{\text{disc}}$
    clip the parameters in $\text{disc}$ (following WGAN)

    # get accuracy loss for domain translator $\xi$
    translate the source batch to get $\xi(U_B)$
    $\text{loss}_{\text{acc}} = -\text{disc}(\xi(U_B)).\text{mean}()$

    # get equivariance loss for domain translator $\xi$
    translate the source batch to get $UU'_B$
    $p_1 = \text{proj}(UU'_B)$
    $p_2 = \text{proj}(\xi(UU'_B))$
    $\text{loss}_{\text{eq}} = -\text{cosine\_similarity}(p_1.\text{detach}(), \text{pred}(p_2)).\text{mean}()$

    # training domain translator $\xi$
    $\text{loss}_{\xi} = \text{loss}_{\text{acc}} + \lambda \cdot \text{loss}_{\text{eq}}$
    update $\xi$ according to $\text{loss}_{\xi}$
  end
end

Evaluation Metrics. We evaluate the trained classifiers with three metrics: the *robust accuracy*, denoted as $R$, measures the probability of a model preserving its output under input variations, the *robust classification accuracy*, denoted as $RC$, measures the probability of a model predicting the correct label under input variations, the *standard accuracy*, denoted as $S$, measures the probability of a model predicting the correct label. During testing, we randomly sample 20 transformed versions for each example to estimate the expectation of robust accuracy and robust classification accuracy.

C.1. Our Method

We use Wasserstein GAN (Arjovsky et al., 2017) to train a domain translator where the inputs of the generator (i.e. domain translator) are source images and the outputs are encouraged to be similar to the target images. We use the encoder-decoder model architecture for implementing the domain translator (i.e. generator), which consists of two convolutional layers for down-sampling, two residual blocks for latent propagation, and two other convolutional layers for up-sampling. The discriminator then distinguishes the real target data from the fake ones translated from the source data. We train generator and discriminator with adversarial training following WGAN where we use 0.01 as the clip value of the discriminator’s weight. For training equivariant domain translator, we use the mean-squared-error (MSE) loss for the equivariance regularization term (the second term in Eq. 4.1). We set $\lambda = 1$ in Eq. 4.1.

---

4Each of them can be viewed as one minus the corresponding loss (instantiated with zero-one loss) defined in Section 3.1.
Algorithm 2 Training the classifier (PyTorch-style pseudocode)

**Input**: trained domain translator $\xi^*$, classifier $f$

- target data $\{x_i\}_{i=1}^N$, source data $\{(u_i, u'_i)\}_{i=1}^M$ (consists of transformed pairs),
- batch size $B$, coefficient $\lambda_1$ and $\lambda_2$ for weighing the trained and the identity domain translator.

**Output**: trained classifier $f^*$

randomly initialize all modules

for epoch in range(max_training_epochs) do

for a target batch $\{(x_i, label_i)\}_{i=1}^B$ in all target data do

randomly sample a source batch $\{(u_i, u'_i)\}_{i=1}^B$

denote $X_B = \{x_i\}_{i=1}^B$, $Label_B = \{label_i\}_{i=1}^B$, $U_B = \{u_i\}_{i=1}^B$, $U'_B = \{u'_i\}_{i=1}^B$,

$UU'_B = \{(u_i, u'_i)\}_{i=1}^B$, $\xi(B) = \{\xi(u_i)\}_{i=1}^B$

# get classification loss

$l_{\text{classify}} = \text{cross\_entropy}(f(X_B), Label_B)$

# get consistency loss under trained domain translator $\xi^*$

translate the source batch to get $\xi^*(U_B)$ and $\xi^*(U'_B)$ # online translation

$p_1 = \text{softmax}(f(\xi^*(U_B)))$

$p_2 = \text{softmax}(f(\xi^*(U'_B)))$

$\text{loss}_{\text{trained}} = \text{kl\_divergence}(p_1, p_2, \text{detach}())$

# get consistency loss under identity map

translate the source batch to get $UU'_B$

$p_1 = \text{softmax}(f(U'_B))$

$p_2 = \text{softmax}(f(U_B))$

$\text{loss}_{\text{identity}} = \text{kl\_divergence}(p_1, p_2, \text{detach}())$

# training classifier $f$

$\text{loss} = \text{loss}_{\text{classify}} + \lambda_1 \cdot \text{loss}_{\text{trained}} + \lambda_2 \cdot \text{loss}_{\text{identity}}$

update $f$ according to $\text{loss}$

end

end

For the robust classifier, we use ResNet18 as the architecture. Since the zero-one loss is difficult to optimize directly, we follow the common practice of using the surrogate loss (Bartlett et al., 2006). We use the cross-entropy loss for training the classifier, including the robustness regularization term $I_1$, similar to Zhang et al. (2019). The MSE loss and the $L^1$ norm loss are two common training objectives that measure the difference between two images in the pixel space. They are used as the reconstruction loss in VAE, CycleGAN, Diffusion Model, etc. We also tried the $L^1$ loss for the equivariance regularization term but did not observe substantial difference. In all our experiments, we use cross-entropy loss as the surrogate loss for training and regularizing the classifier. We set $\lambda_1 = \lambda_2 = 0.5$ in Eq. 4.2. Since accurately estimating the $W_1$ distance for multi-dimensional non-Gaussian distributions is difficult, we use the Fréchet inception distance (FID, see Heusel et al. (2017)) to evaluate how well the domain translator pushes forward the source data to approximate the target data.

C.2. MBRDL

MBRDL (model-based robust deep learning, (Robey et al., 2020)) learns a model to simulate the natural variation. In their paper, the variation model is learned and applied to the same domain. Their method can easily extend to scenarios where variations are unforeseen in the target domain but is available in the source domain. In this paper, we first learn a variation simulator with the source data where transformed pairs are used for learning variations. We use MUNIT (Huang et al., 2018) as the variation simulator following settings in (Robey et al., 2020). MUNIT is first designed for style transfer, here (Robey
et al., 2020) use it for input transformation. Then, we apply the variation simulator directly to the target data to do data variation and train robust classifiers with a consistency regularization loss addition to the classification loss.

C.3. UDA

UDA (unsupervised data augmentation, (Xie et al., 2020)) improves the model’s robustness against variations with consistency regularization on unlabeled data. Although the unlabeled data is very similar to the target data and has foreseen variations in their paper, we can directly use their method in our case. We see source data as the unlabeled data and do consistency regularization on it while training the classifier on the target data. It’s easy to see that, UDA is a simple version of our method where $\lambda_1 = 0$ in Eq. 4.2. In our experiments of UDA, we set $\lambda_2 = 1$.

D. Additional Results

D.1. Visualizing the Results of Equivariant Domain Translator

We show the outputs of our domain translators in Figure 10, 11 and 12. Results demonstrate that our method can effectively translate the source data to be target-like. The trained domain translator also well-preserve the variations including random rotation, RandAugment, and 3D-viewpoint change. Therefore, we are able to do consistency regularization with the target-like images and the transformed version of them, so that to train a robust classifier under unforeseen variations. We notice that domain translators trained with different source dataset have different performances. As discussed in Section 5.3, the source dataset’s distance to the target dataset correlates with the performance. Additionally, if the source dataset is much “simpler” than the target one, such as MNIST and SVHN, it is very difficult for the domain translator to cover the whole manifold of the target distribution, and to preserve complex variations such as RandAugment (especially the color change) on MNIST. One interesting future work is to take the intrinsic dimension of the dataset into consideration.

In addition, we evaluate the capability of the equivariant domain translator to preserve more real-world variations using limited data. To this end, we train the equivariant domain translator to preserve illumination changes from the Multi-illumination dataset (comprising 1015 images, Murmann et al. (2019)) to the labeled training set of STL-10 (comprising 5k images). Figure 9 shows some domain-translated images for visual evaluation.

![Figure 9. Visualization of some domain-translated images from the Multi-illumination dataset (comprising 1015 images) to the STL10’s training set (comprising 5k images), demonstrating the preservation of illumination changes using limited data.](image)

D.2. Results on CIFAR-100

Table 6 shows the results on CIFAR-100 where we use SVHN, STL10 and CIFAR-10 as the source data. Data variation is the RandAugment. We get consistent results where our method excels over other methods in robustness and accuracy.

D.3. Additional Baselines

Incorporating additional baselines, although not originally designed to address our specific problem, can stimulate deeper insights into the problem setting. In Table 7, we consider contrastive self-supervised learning and additionally evaluate 1)
Learning Unforeseen Robustness from Out-of-distribution Data Using Equivariant Domain Translator

(a) SVHN as the source dataset. Random rotation as the variation.

(b) STL10 as the source dataset. Random rotation as the variation.

Figure 10. Results of our method with random rotation as the input variation. We use CIFAR-10 as the target dataset. \( z \) denotes the source data, \( \phi \) denotes the variation, i.e. random rotation, and \( \xi \) denotes Eq, the domain translator trained with the heuristic method. By comparing \( \xi(z) \) with CIFAR-10 data, results indicate that our method can effectively translate the source data to be target-like. By comparing between \( \xi \circ \phi(z) \) and \( \phi \circ \xi(z) \), which are expected to be similar, our domain translators well-preserve the variations.

D.4. Sensitivities to Source Sample Size

We further investigate whether the number of source datapoints is a significant factor. In Table 8 we show some initial results. Our empirical findings suggest that 1) when the source sample size is "comparable" with the target sample size (greater than 25k or 50% of the target training sample size), there is no noticeable change in the final robust classification accuracy. 2) As the source sample size decreases below 50% of the target sample size, the robust classification accuracy gradually drops. Nonetheless, our method still offers a small benefit over ERM even when the source sample size is as small as one batch size (256).

D.5. Problems of MBRDL

Figure 13 and 14 shows the performance of the variation simulator learned by MBRDL. We can see that the MBRDL suffers from two problems. Firstly, it is hard to learn a good variation simulator. As (Zhou et al., 2022) observed and as shown in...
Figure 11. Results of our method with RandAugment as the input variation. We use CIFAR-10 as the target dataset. $z$ denotes the source data, $\phi$ denotes the variation, i.e. RandAugment, and $\xi$ denotes Eq. the domain translator trained with the heuristic method. By comparing $\xi(z)$ with CIFAR-10 data, results indicate that our method can effectively translate the source data to be target-like. By comparing between $\xi \circ \phi(z)$ and $\phi \circ \xi(z)$, which are expected to be similar, our domain translators well-preserve the variations in most cases.
Learning Unforeseen Robustness from Out-of-distribution Data Using Equivariant Domain Translator

**Figure 12.** Results of our method with 3D-viewpoint change. We use CIFAR-10 as the target dataset and Objectron as the source dataset. Here, \( z \) denotes the source data, \( \phi \) denotes the variation, i.e. 3D-viewpoint change, and \( \xi \) denotes Eq, the domain translator trained with the heuristic method. By comparing \( \xi(z) \) with CIFAR-10 data, results indicate that our method can effectively translate the source data to be target-like. \( \xi \circ \phi(z) \) shows that the domain translator well-preserves the 3D-viewpoint change. For example, in the fourth column, two cars generated by \( \xi(z) \) and \( \xi \circ \phi(z) \) well-preserve the viewpoint change that exists in two chair images (i.e. \( z \) and \( \phi(z) \)).

**Table 6.** Results of classifiers trained using different methods and source datasets. The target dataset is CIFAR-100 w/o data augmentation and the data variation is RandAugment. We show here for reference the oracle method that does consistency regularization directly on the target dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Src</th>
<th>( R_C ) (%)</th>
<th>( R ) (%)</th>
<th>( S ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>/</td>
<td>48.8 ± 0.1</td>
<td>57.2 ± 0.2</td>
<td>62.9 ± 0.3</td>
</tr>
<tr>
<td>MBRDL</td>
<td>SVHN</td>
<td>36.9 ± 0.4</td>
<td>55.3 ± 0.5</td>
<td>52.4 ± 0.3</td>
</tr>
<tr>
<td>UDA</td>
<td>SVHN</td>
<td>51.7 ± 0.2</td>
<td>61.6 ± 0.2</td>
<td>63.2 ± 0.4</td>
</tr>
<tr>
<td>EDT (Ours)</td>
<td>SVHN</td>
<td>53.2 ± 0.3</td>
<td>63.4 ± 0.2</td>
<td>64.1 ± 0.3</td>
</tr>
<tr>
<td>MBRDL</td>
<td>STL10</td>
<td>39.6 ± 0.3</td>
<td>56.1 ± 0.3</td>
<td>56.1 ± 0.2</td>
</tr>
<tr>
<td>UDA</td>
<td>STL10</td>
<td>55.9 ± 0.3</td>
<td>67.1 ± 0.2</td>
<td>64.1 ± 0.3</td>
</tr>
<tr>
<td>EDT (Ours)</td>
<td>STL10</td>
<td>58.3 ± 0.3</td>
<td>70.0 ± 0.3</td>
<td>65.1 ± 0.3</td>
</tr>
<tr>
<td>MBRDL</td>
<td>CIFAR-10</td>
<td>39.6 ± 0.4</td>
<td>58.4 ± 0.3</td>
<td>56.2 ± 0.3</td>
</tr>
<tr>
<td>UDA</td>
<td>CIFAR-10</td>
<td>56.5 ± 0.2</td>
<td>68.3 ± 0.2</td>
<td>63.8 ± 0.3</td>
</tr>
<tr>
<td>EDT (Ours)</td>
<td>CIFAR-10</td>
<td>59.0 ± 0.2</td>
<td>71.2 ± 0.3</td>
<td>64.5 ± 0.2</td>
</tr>
<tr>
<td>Oracle</td>
<td>/</td>
<td>70.9 ± 0.2</td>
<td>82.1 ± 0.2</td>
<td>73.6 ± 0.2</td>
</tr>
</tbody>
</table>

**Figure 13.** brightness change and color change are easy to learn but geometric transformations such as rotation are hard to learn. The complex variations such as RandAugment are even harder. Secondly, the learned variation simulator has poor generalization ability. Figure 14 (a) and (c) show that the variation simulator which is trained on the source data performs well on the source data. However, (b) and (d) show that the variation simulator performs badly when directly applied to the target data, resulting in blurred images or content-changed images. We suspect that it is because the variation is very hard to learn and it is even harder to learn a variation simulator that is disentangled from the source data. The problems get severe when the target domain and the source domain are far from each other. This explains why MBRDL hurts the robustness and accuracy in our experiments.
Table 7. Performance of SimCLR on SVHN and STL10 datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>Source</th>
<th>Weight</th>
<th>Robust Classification Accuracy</th>
<th>Robust Accuracy</th>
<th>Standard Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR</td>
<td>SVHN</td>
<td>0.001</td>
<td>81.2 ± 0.2</td>
<td>84.7 ± 0.2</td>
<td>89.4 ± 0.1</td>
</tr>
<tr>
<td>SimCLR</td>
<td>SVHN</td>
<td>0.01</td>
<td><strong>81.8 ± 0.1</strong></td>
<td><strong>85.5 ± 0.1</strong></td>
<td><strong>89.6 ± 0.2</strong></td>
</tr>
<tr>
<td>SimCLR</td>
<td>SVHN</td>
<td>0.1</td>
<td>80.8 ± 0.1</td>
<td>84.6 ± 0.1</td>
<td>89.5 ± 0.3</td>
</tr>
<tr>
<td>SimCLR</td>
<td>SVHN</td>
<td>1</td>
<td>80.4 ± 0.1</td>
<td>84.4 ± 0.2</td>
<td>89.1 ± 0.0</td>
</tr>
<tr>
<td>SimCLR</td>
<td>STL10</td>
<td>0.001</td>
<td>79.7 ± 0.2</td>
<td>82.8 ± 0.3</td>
<td>89.4 ± 0.0</td>
</tr>
<tr>
<td>SimCLR</td>
<td>STL10</td>
<td>0.01</td>
<td>82.1 ± 0.2</td>
<td>85.4 ± 0.3</td>
<td>89.6 ± 0.1</td>
</tr>
<tr>
<td>SimCLR</td>
<td>STL10</td>
<td>0.1</td>
<td><strong>84.0 ± 0.1</strong></td>
<td><strong>87.8 ± 0.1</strong></td>
<td><strong>90.3 ± 0.4</strong></td>
</tr>
<tr>
<td>SimCLR</td>
<td>STL10</td>
<td>1</td>
<td>81.6 ± 0.8</td>
<td>85.5 ± 0.9</td>
<td>89.8 ± 0.3</td>
</tr>
</tbody>
</table>

Table 8. Performance on SVHN and STL10 datasets with varying training sizes.

<table>
<thead>
<tr>
<th>src</th>
<th>0 (ERM)</th>
<th>256</th>
<th>1,024</th>
<th>4,096</th>
<th>16,384</th>
<th>65,536</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVHN</td>
<td>79.1 ± 0.2</td>
<td>80.1 ± 0.4</td>
<td>80.2 ± 0.5</td>
<td>80.8 ± 0.3</td>
<td>82.6 ± 0.3</td>
<td>-</td>
<td>83.2 ± 0.3 (73,257)</td>
</tr>
<tr>
<td>STL10</td>
<td>79.1 ± 0.2</td>
<td>81.0 ± 0.5</td>
<td>81.7 ± 0.4</td>
<td>82.0 ± 0.3</td>
<td>84.8 ± 0.2</td>
<td>87.9 ± 0.3</td>
<td>87.8 ± 0.2 (100,000)</td>
</tr>
</tbody>
</table>

(a) Apply rotation simulator learned on SVHN to SVHN.

(b) Apply rotation simulator learned on SVHN to CIFAR10.

(c) Apply rotation simulator learned on STL10 to STL10.

(d) Apply rotation simulator learned on STL10 to CIFAR10.

Figure 13. Results of MBRDL with random rotation as the input variation. In every subfigure, the first line shows the original images and the second line shows the transformed ones using the learned variation simulator.
(a) Apply RandAugment simulator learned on SVHN to SVHN.

(b) Apply RandAugment simulator learned on SVHN to CIFAR10.

(c) Apply RandAugment simulator learned on STL10 to STL10.

(d) Apply RandAugment simulator learned on STL10 to CIFAR10.

Figure 14. Results of MBRDL with RandAugment as the input variation. In every subfigure, the first line shows the original images and the second line shows the transformed ones using the learned variation simulator.