DC4L: Distribution Shift Recovery via Data-Driven Control for Deep Learning Models

Vivian Lin VILIN@SEAS.UPENN.EDU
Kuk Jin Jang JANGKJ@SEAS.UPENN.EDU
Souradeep Dutta DUTTASO@SEAS.UPENN.EDU
Michele Caprio CAPRIO@SEAS.UPENN.EDU
Oleg Sokolsky SOKOLSKY@SEAS.UPENN.EDU
Insup Lee LEE@SEAS.UPENN.EDU
University of Pennsylvania

Abstract

Deep neural networks have repeatedly been shown to be non-robust to the uncertainties of the real world, even to naturally occurring ones. A vast majority of current approaches have focused on data-augmentation methods to expand the range of perturbations that the classifier is exposed to while training. A relatively unexplored avenue that is equally promising involves sanitizing an image as a preprocessing step, depending on the nature of perturbation. In this paper, we propose to use control for learned models to recover from distribution shifts online. Specifically, our method applies a sequence of semantic-preserving transformations to bring the shifted data closer in distribution to the training set, as measured by the Wasserstein distance. Our approach is to 1) formulate the problem of distribution shift recovery as a Markov decision process, which we solve using reinforcement learning, 2) identify a minimum condition on the data for our method to be applied, which we check online using a binary classifier, and 3) employ dimensionality reduction through orthonormal projection to aid in our estimates of the Wasserstein distance. We provide theoretical evidence that orthonormal projection preserves characteristics of the data at the distributional level. We apply our distribution shift recovery approach to the ImageNet-C benchmark for distribution shifts, demonstrating an improvement in average accuracy of up to 14.21% across a variety of state-of-the-art ImageNet classifiers. We further show that our method generalizes to composites of shifts from the ImageNet-C benchmark, achieving improvements in average accuracy of up to 9.81%. Finally, we test our method on CIFAR-100-C and report improvements of up to 8.25%.

Keywords: distribution shift, Markov decision process, reinforcement learning

1. Introduction

Deep learning models are excellent at learning patterns in large high dimensional datasets. However, the brittleness of deep neural networks (DNNs) to distribution shifts is a challenging problem. Hendrycks and Gimpel (2018) showed that, even in naturally occurring distribution shift scenarios, a classifier's performance can deteriorate substantially. Arguably, one of the most widespread uses of deep learning techniques currently is in image recognition. In this paper, we propose to use decision and control for learned models (DC4L) to improve robustness to distribution shifts, with demonstration on image classification tasks. The idea is to actively *sanitize* a set of images depending on the type of the distribution shift. Intuitively, the training distribution of a classifier is viewed as its *comfort zone*, where the behavior is more predictable and trustworthy. At run time, when exposed to perturbations that push the images outside of this comfort zone, a feedback policy takes control actions which can bring the images back to a more familiar space. The control actions are so chosen that the semantic meaning of the images are preserved. This ensures correctness. While approaches for DNN robustness have until now focused on data-augmentation techniques (Hendrycks et al., 2019; Erichson et al., 2022; Verma et al., 2019; Yun et al., 2019; Kim et al., 2020b; Hendrycks et al., 2020), we show that by using ideas from the data-driven control paradigm, it is possible to provide an additional level of performance boost beyond what can be offered by SOTA augmentation methods.

Our technique exploits the following observation: when distribution shift arises in the external environment due to natural causes, it persists for a certain duration of time. For instance, when a corruption in image quality occurs due to snow, this corruption does not disappear in the next image frame. This gives the system some time to *adapt* and *recover* from this shift by computing some semantic preserving transformations to the data. Our technique, Supervisory system for Shift Adaptation and Recovery (`SuperStAR`), applies a sequence of semantic-preserving transforms to the input data, correcting the input to align with the original training set of the classifier. We show that formulating the sequence selection problem as a Markov decision process (MDP) lends a natural solution: reinforcement learning (RL).

In summary, our contributions towards addressing the problem of robustness to semantic preserving shifts are as follows. 1) We translate the problem of distribution shift recovery for neural networks to a Markov decision process, which we solve using reinforcement learning. 2) We identify a minimum condition of operability for our method, which we check online using a binary classifier. 3) We develop a method to efficiently compute the degree of distribution shift by projecting to a lower dimensional space. This uses results from Cai and Lim (2022) in conjunction with the Wasserstein distance. 4) We demonstrate an application to ImageNet-C and achieve significant accuracy improvements (up to 14.21% averaged across all shift severity levels) on top of standard training and data-augmentation schemes. We further show that our method generalizes beyond the ImageNet-C benchmark, yielding up to 9.81% on composite Imagenet-C shifts and up to 8.25% on CIFAR-100-C in accuracy improvements.

2. Preliminaries

We begin by assuming that the images are sampled from a measurable space $(\mathcal{X}, \mathcal{A}_\mathcal{X})$. Let $\Delta(\mathcal{X}, \mathcal{A}_\mathcal{X})$ denote the set of all probability measures on $(\mathcal{X}, \mathcal{A}_\mathcal{X})$. We pick a distribution $D \in \Delta(\mathcal{X}, \mathcal{A}_\mathcal{X})$ from which the current set of images are sampled. Assume that the labels belong to a measurable space $(\mathcal{Y}, \mathcal{A}_\mathcal{Y})$, and a classifier C is an $\mathcal{A}_\mathcal{X} \backslash \mathcal{A}_\mathcal{Y}$ measurable map, $\mathcal{C} : \mathcal{X} \to \mathcal{Y}$. An *oracle classifier* $\mathcal{C}^*$ produces the ground truth labels. Next, we define a semantic preserving transform $\mathbb{T}$.

Definition 1 (Semantic Preserving Transform) *A function $\mathbb{T} : \mathcal{X} \to \mathcal{X}$ is semantic preserving iff $\mathcal{C}^*(x) = \mathcal{C}^*(\mathbb{T}(x))$, for all $x \in \mathcal{X}$.*

We denote by $\mathbb{S}$ the set of all such semantic preserving transforms. In the standard empirical risk minimization (ERM) paradigm, we approximate $\mathcal{C}^*$ with some classifier $\mathcal{C}$. When measuring robustness to common corruptions, typically a corrupting transform $\mathbb{T}_c$ belongs to $\mathbb{S}$. This includes transforms like the addition of Gaussian noise, speckle noise, and alike. The error of a classifier $\mathcal{C}$ under the distribution D, is defined as

$$err(D) := \mathbb{E}_{x \sim D} \left[\mathbb{1}(\mathcal{C}^*(x) \neq \mathcal{C}(x)) \right],$$

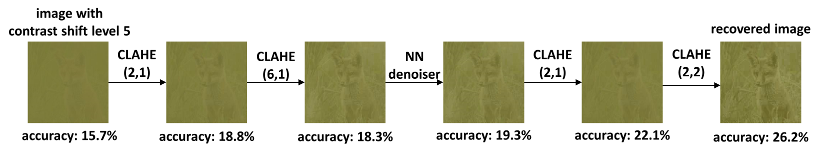

Figure 2: Example transformations applied to an image with contrast shift level 5. CLAHE(x,y) denotes histogram equalization with strength determined by x and y (details in Appendix E of Lin et al. (2023)). The policy applies a non-trivial composition of transformations that would be difficult to find through manual manipulation. The policy chooses few redundant actions and improves the accuracy of an AugMix-trained ResNet-50 on a random batch of 1000 images.

where $\mathbb{1}$ is the standard indicator function, which evaluates to 1 iff $\mathcal{C}^*(x) \neq \mathcal{C}(x)$.[1] For a robust classifier we expect $err(D)$ to be minimal for multiple choices of the distribution D. For instance in the case of common corruptions introduced in Hendrycks and Dietterich (2019), the goal is to optimize the choice of classifier $\mathcal{C}$ such that $err(D)$ is minimized, even under shift. This is typically achieved using data-augmentation schemes such as Augmix, NoisyMix, and DeepAugment.

3. Functionality and Problem Statement

At a high level, the functioning of `SuperStAR` is akin to a supervisor for the classifier shown in Figure 1. `SuperStAR` detects distribution shifts and computes a recovery strategy to be applied before sending an image to a classifier. Determining the appropriate recovery strategy presents an interesting challenge.

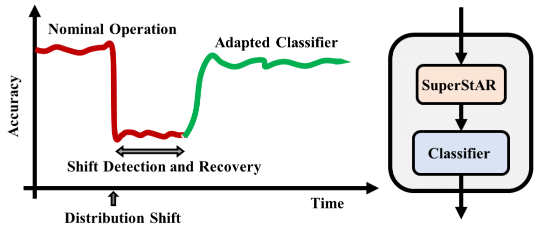

Figure 1: Overview of `SuperStAR`. At deployment, assume that a distribution shift causes a drop in accuracy. This is detected through changes in the Wasserstein distance between a validation set and the corrupted set. `SuperStAR` computes a composition of transforms $\mathcal{I}_k$ to adapt to the shift and recover accuracy. This composition of `SuperStAR` with the classifier helps it detect and adapt, boosting robustness of classification.

Consider a random variable $x_c = \mathbb{T}(z)$, where $z \sim D$ and $\mathbb{T}$ is a semantic preserving transform. Note that $\mathbb{T}$ is generally not invertible. However, one may possibly choose an element $\mathbb{T}'$ from the set of semantic preserving transforms $\mathbb{S}$ that partially recovers the accuracy drop due to $\mathbb{T}$. It might even be effective to select a sequence of such transforms: an ordered set $\mathcal{T} := \{\mathbb{T}_k, \mathbb{T}_{k-1}, \ldots \mathbb{T}_1\}$ with $k \geq 1$, such that $\mathbb{S} \ni \mathcal{I}_k(x) := \mathbb{T}_k \circ \mathbb{T}_{k-1} \circ \cdots \circ \mathbb{T}_1(x)$.[2] Ideally, we wish to find an algorithm which optimizes the following:

$$\mathcal{I}_k^* = \text{argmin}_{\mathcal{I}_k} \; R(\mathcal{I}_k) \text{ , where } R(\mathcal{I}_k) = err(D_{\mathcal{I}_k \circ \mathbb{T}}) - err(D), \tag{1}$$

with $D_{\mathcal{I}_k \circ \mathbb{T}}$ denoting the distribution of $\mathcal{I}_k \circ \mathbb{T}(z)$, $z \sim D$. The transform $\mathcal{I}_k$ is what effectively *reverses* an image corruption due to $\mathbb{T}$.

At deployment, when `SuperStAR` detects a shift in distribution compared to a clean validation set, it should propose a composition of transforms $\mathcal{I}_k$ to minimize the cost outlined in Equation 1. From the vantage point of classifier $\mathcal{C}$, images transformed using $\mathcal{I}_k$ *appear* to be closer to the *home* distribution D. An example result of this is shown in Figure 2.

4. Distribution Shift Recovery is a Markov Decision Process

Our task is to compute a composition of transforms $\mathcal{I}_k$ to apply to the corrupted set $\mathbf{V}_c$, sampled i.i.d from $D_{\mathbb{T}}$, to realize the optimization cost outlined in Equation 1. To this end, we note the following theorem.

1. For notational convenience, we hereafter denote both a random variable and its realization by a lowercase Latin letter.
2. The order $\prec_{\mathcal{T}}$ on $\mathcal{T}$ is given by $\mathbb{T}_i \prec_{\mathcal{T}} \mathbb{T}_j \iff i < j, i, j \in \mathbb{N}$. $\mathbb{T}_0$ is assumed to be the identity function.

Theorem 2 *$R(\mathcal{I}_k) \leq \alpha \cdot d_{TV}(D, D_{\mathcal{I}_k \circ \mathbb{T}})$, for some finite $\alpha \in \mathbb{R}$ and semantic preserving transform $\mathcal{I}_k$.*

Proof Let us denote by d the pdf of $D_{\mathcal{I}_k \circ \mathbb{T}}$ and by d' the pdf of D. We begin by expanding the definition of $R(\mathcal{I}_k)$.

$$\begin{aligned}
R(\mathcal{I}_k) &= err(D_{\mathcal{I}_k \circ \mathbb{T}}) - err(D) \\
&= \mathbb{E}_{x_1 \sim D_{\mathcal{I}_k \circ \mathbb{T}}, x_2 \sim D}\left[\mathbb{1}(\mathcal{C}^*(x_1) \neq \mathcal{C}(x_1)) - \mathbb{1}(\mathcal{C}^*(x_2) \neq \mathcal{C}(x_2))\right] \\
&= \int_{\mathcal{X}} \mathbb{1}(\mathcal{C}^*(x) \neq \mathcal{C}(x))[d(x) - d'(x)]\mathrm{d}x \\
&\leq \left|\int_{\mathcal{X}} \left(\sup_{x \in \mathcal{X}} \mathbb{1}(\mathcal{C}^*(x) \neq \mathcal{C}(x))\right)[d(x) - d'(x)]\mathrm{d}x\right| \\
&= \left|\left(\sup_{x \in \mathcal{X}} \mathbb{1}(\mathcal{C}^*(x) \neq \mathcal{C}(x))\right) \int_{\mathcal{X}}[d(x) - d'(x)]\mathrm{d}x\right| \\
&\leq \left|\sup_{x \in \mathcal{X}} \mathbb{1}(\mathcal{C}^*(x) \neq \mathcal{C}(x))\right| \cdot \sup_{A \in \mathcal{A}_{\mathcal{X}}} \left|\int_A [d(x) - d'(x)]\mathrm{d}x\right| \\
&= \alpha \cdot d_{TV}(D, D_{\mathcal{I}_k \circ \mathbb{T}}),
\end{aligned}$$

where $\alpha = |\sup_{x \in \mathcal{X}} \mathbb{1}(\mathcal{C}^*(x) \neq \mathcal{C}(x))|$. The first step assumes $\mathcal{X}$ is a continuous space. The rest follows from expressing the definition of computing expectation in terms of the classifier error. ■

Thus, for a transform $\mathbb{T}$ (possibly a corruption), it is possible for the classifier to recover performance if the apparent distribution under $\mathcal{I}_k \circ \mathbb{T}$ is close enough to the original distribution D. Then the optimal $\mathcal{I}_k$ is the sequence of semantic preserving transforms that minimize the distance from D. Equivalently, this can be viewed as a sequence of actions maximizing a reward. More formally, the task of computing $\mathcal{I}_k$ can be formulated as a reactive policy for an MDP, which can be learned using standard reinforcement learning techniques. In this section, we present this MDP formulation in the context of an image classification task, but it may be applied in any learning setting.

Definition 3 (MDP) *A* Markov Decision Process (MDP) *is a 6-tuple $\mathcal{E} = (\mathcal{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathrm{I}_0)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is the set of states, $\mathbb{A} \subseteq \mathbb{R}^m$ is the set of actions, $\mathcal{P}(s'|s, a)$ specifies the probability of transitioning from state s to s' on action a, $\mathcal{R}(s, a)$ is the reward returned when taking action a from state s, $\gamma \in [0, 1)$ is the discount factor, and I_0 is the initial state distribution.*

State Representation. The environment has access to a set $\mathbf{V}_c \subset \mathcal{X}$, which is a set of possibly corrupted images. A state of the MDP is a compressed representation of this set $\mathbf{V}_c$, capturing the type of corruptions in an image. Let us assume that this projection is captured by some function $\mathbb{F}_R : \mathcal{A}_{\mathcal{X}} \to \mathfrak{R}$, where $\mathfrak{R}$ is the space of representations for a set of images. We want a representation $r = \mathbb{F}_R(\mathbf{V}_c)$ to be rich enough that a policy can decipher the appropriate choice of action in $\mathbb{A}$, but also compact enough that it is possible to learn a policy within a few episodes. Typically, a smaller state space size leads to faster convergence for reinforcement learning algorithms.

In this paper, for a set of images $\mathbf{V}_c$, we select a 3-dimensional state representation that measures the average brightness, standard deviation, and entropy of the images in $\mathbf{V}_c$. We convert each image x to grayscale, then obtain the discrete wavelet transform and compute its average brightness $B(x)$, standard deviation $S(x)$, and entropy $E(x)$. The state representation is an average of all these values across the images,

$$\mathbb{F}_R(\mathbf{V}_c) = \frac{1}{|\mathbf{V}_c|} \sum_{x \in \mathbf{V}_c} \begin{bmatrix} B(x), & S(x), & E(x) \end{bmatrix}^\top. \tag{2}$$

Actions and Transitions. The set of actions $\mathbb{A} \subseteq \mathbb{S}$ is a set of semantic preserving transforms from which the learner chooses to maximize some reward. For an example, see the action set selected for the ImageNet-C benchmark in Appendix E of Lin et al. (2023). Hence, capturing Equation 1 as a reward leads the agent to pick actions that mitigate the current corruption to some extent. Transitions model the effect of applying a transform from $\mathbb{S}$ to a possibly corrupted set $\mathbf{V}_c$. With slight abuse of notation, we use $\mathbb{T}(\mathbf{V}_c)$ to denote set $\{v' : v' = \mathbb{T}(v), v \in \mathbf{V}_c\}$.

Computing Reward. As shown in Figure 3, computing the reward for a set of images $\mathbf{V}_c$ corresponds to measuring the distance from a clean validation set $\mathbf{V}$. Ideally, the distance between the distributions from which $\mathbf{V}_c$ and $\mathbf{V}$ are drawn would be measured, but this is difficult without knowledge of the source distributions. Instead, we use an empirical estimate of the Wasserstein distance (Bonneel et al., 2011) to compute the distributional distance between sets $\mathbf{V}_c$ and $\mathbf{V}$. It should be noted that a variety of distance functions can be employed here. We explore alternatives in Appendix A of Lin et al. (2023).

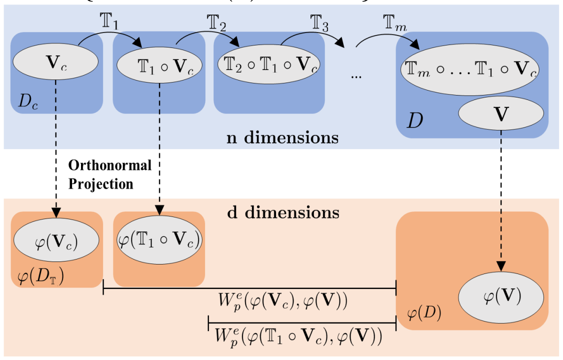
$\mathbb{T}_1$
$\mathbb{T}_2$
$\mathbb{T}_3$
$\mathbb{T}_m$
$\mathbf{V}_c$
$\mathbb{T}_1 \circ \mathbf{V}_c$
$\mathbb{T}_2 \circ \mathbb{T}_1 \circ \mathbf{V}_c$
...
$\mathbb{T}_m \circ \ldots \mathbb{T}_1 \circ \mathbf{V}_c$
D_c
D
$\mathbf{V}$
n dimensions
Orthonormal Projection
d dimensions
$\varphi(\mathbf{V}_c)$
$\varphi(\mathbb{T}_1 \circ \mathbf{V}_c)$
$\varphi(D_{\mathbb{T}})$
$W_p^e(\varphi(\mathbf{V}_c), \varphi(\mathbf{V}))$
$W_p^e(\varphi(\mathbb{T}_1 \circ \mathbf{V}_c), \varphi(\mathbf{V}))$
$\varphi(D)$
$\varphi(\mathbf{V})$

Figure 3: Operation of SuperStAR. Starting from $\mathbf{V}_c$, the algorithm selects a sequence of transforms $\mathcal{T}$ which move $\mathbf{V}_c$ closer to the original distribution $\mathbf{V}$. During the sequence, the orthonormal projections $\varphi(\mathbf{V})$ and $\varphi(\mathbf{V}_c)$ are used to compute the Wasserstein distance $W_p(\varphi(\mathbf{V}_c), \varphi(\mathbf{V}))$. See Section 6 for details.

In practice the policy might not be able to reduce the W_p^e to a level such that the classifier completely recovers the loss in accuracy. One reason for this is the possible non-existence of the inverse of the corruption transform. Another possible issue is that a transformation may overly alter the image such that the classifier performs poorly. Although we can combat against this by ensuring that actions make incremental changes to the image, this is hard to control. We therefore add a regularizer to the Wasserstein distance that penalizes excessive changes to the image. This is achieved using a visual similarity between pairs of images known as *ssim* (Wang et al., 2004).

Given $\lambda > 0$ and $0 \leq \omega < 1$, the reward function is given by

$$R(s_t, a_t) = -W_p^e(\mathbf{V}, \mathbb{F}_R^{-1}(s_t)) + \lambda L_S(\mathbb{F}_R^{-1}(s_0), \mathbb{F}_R^{-1}(s_{t+1})), \tag{3}$$

where,

$$L_S(\mathbf{X}, \mathbf{Y}) = \begin{cases} \log(1 - ssim(\mathbf{X}, \mathbf{Y})), & ssim(\mathbf{X}, \mathbf{Y}) < \omega \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Note that the regularization hyperparameter λ influences the level of aggression in correction. For example, selecting a large λ favors actions with minimal influence on the data.

Initial State. At test time, the initial state of the MDP is produced by a random environment corruption from the set $\mathbb{S}$ that the system is subjected to. In reality the designer does not have access to any of these corrupting transforms. Hence, at training time we train a policy network to reverse a set of *surrogate* corruptions from $\mathbb{S}$, with the hope that some of these transfer at inference time to an unseen set of corruptions. An example surrogate corruption set and selection methodology for the ImageNet-C benchmark is given in Appendix E of Lin et al. (2023). We pick uniformly randomly from a finite set of $\mathsf{S}_c \subset \mathbb{S}$ of these surrogate corruptions to sample the initial state I_0.

Standard RL techniques can be applied to learn a policy $\pi : \mathcal{S} \to \mathbb{A}$ for the MDP $\mathcal{E}$, which is a strategy to recover the distribution shift. The value of a state for policy π is the expected return $\mathbf{R}_\pi(s_0)$, starting from state s_0, while executing the policy π at every step, $a_t = \pi(s_t)$. The optimal policy π^* maximizes the reward starting from the initial state distribution – i.e., $\pi^* = \arg\max V^{\mathrm{I}_0}(\pi)$, where $V^{\mathrm{I}_0}(\pi) = \mathbb{E}_{s_0 \in \mathrm{I}_0}[\mathbf{R}_\pi(s_0)]$.

5. Applying a Learned Policy for Detection and Correction

As described in Section 4, a DNN policy π is trained at design time such that, given a corrupted set $\mathbf{V}_c$ of i.i.d observations from corrupted distribution $D_{\mathbb{T}}$, the policy generates a finite composition $\mathcal{I}_k$ of $k < \infty$ transforms from $\mathbb{S}$. This solves the optimization problem in Equation 1. At design time, we assume the algorithm has access to a validation set $\mathbf{V} = \{v_1, \ldots, v_n\}$, where v_i is drawn i.i.d from the training distribution D.

At inference / run-time SuperStAR uses the corrupted set $\mathbf{V}_c = \{v_1^c, v_2^c, \ldots, v_n^c\}$ drawn i.i.d from $D_{\mathbb{T}}$, $\mathbb{T} \in \mathbb{S}$, and the policy π to select a correcting sequence $\mathcal{T} := \{\mathbb{T}_k, \mathbb{T}_{k-1}, \ldots \mathbb{T}_1\}$. We fix a maximum horizon k to keep the algorithm tractable at both learning and deployment. The procedure for selecting sequence $\mathcal{T}$ is presented in Algorithm 1.

Algorithm 1 Transformation Selection

```
Require: Validation set $\mathbf{V}$, corrupted set $\mathbf{V}_c$, policy $\pi$, horizon $k$, thresholds $\alpha, \beta \in (0, 1]$
1:  $w_0 \leftarrow \tilde{W}(\mathbf{V}, \mathbf{V}_c)$
2:  $\mathbf{V}_0 \leftarrow \mathbf{V}_c$
3:  $T \leftarrow \{\ \}$
4:  for $i = 1, \ldots, k$ do
5:      if $\mathbf{V}_i$ is inoperable then
6:          return $T$
7:      end if
8:      $\mathbb{T}_i \leftarrow \pi(\mathbf{V}_{i-1})$
9:      $\mathbf{V}_i \leftarrow \mathbb{T}_i(\mathbf{V}_{i-1})$
10:     $w_i \leftarrow \tilde{W}(\mathbf{V}, \mathbf{V}_i)$
11:     if $w_i \leq \alpha w_0 \vee w_i \geq \beta w_{i-1}$ then
12:         return $T$
13:     end if
14:     $T \leftarrow \{\mathbb{T}_i\} \cup T$
15: end for
16: Return $T$
```

In Line 1, Algorithm 1 uses the estimator $\tilde{W}$ to estimate the initial Wasserstein distance. Next, it runs policy π for k steps (Lines 5−16), where it iteratively applies the transform picked by the policy to update the corrupted set. The algorithm uses a lower dimensional state representation of $\mathbf{V}$ for evaluating the policy and for the estimator $\tilde{W}$ (see Section 4). The algorithm collects and returns this set of transforms in T.

There are two stopping criteria to prevent the selection of damaging transforms. The first, in line 5, checks for a minimum condition of operability on the set (see Section 5.1). The second, in Line 11, guards against overly transforming images to diminishing returns. It also hedges against the chance that the distribution shift is not semantic preserving, which is always possible in reality. Hence, we pause the policy when the Wasserstein distance decreases beyond a threshold.

5.1. Minimum Condition for Operability

The optimal policy π^* selects transformations that maximize the reward function, consisting of some distance function (e.g., the Wasserstein distance) plus a regularizing factor. The efficacy of such a policy thus depends on the distance function being a reliable estimate of a classifier's performance on the given data. We define an operable corrupted set $\mathbf{V}_c$ as follows.

Definition 4 (Operable Set) *For some distance function d, classifier f, validation set $\mathbf{V}$, and set of transformations $\{\mathbb{T}_i\}_{i=1}^t$, a set $\mathbf{V}_c = \{v_1^c, c_2^c, \ldots, v_n^c\}$ is operable iff $\{d(\mathbf{V}, \mathbb{T}_i(\mathbf{V}_c))\}_{i=1}^t$ is negatively linearly correlated with $\{\frac{1}{n}\sum_{j=1}^n \mathbb{1}[f(\mathbb{T}_i(v_j^c)) = \mathcal{C}^*(\mathbb{T}_i(v_j^c))]\}_{i=1}^t$.*

Since the accuracy of the classifier we wish to adapt is unknown on inference-time data, operability is in practice difficult to evaluate. However, we can instead predict operability from the state representation $\mathbb{F}_R(\mathbf{V}_c)$ using a simple binary classifier, which is trained on surrogate corrupted data generated at design time. Label generation can be performed by evaluating the distance function and accuracies on these surrogate shifts, where $\mathbb{T}_i(\mathbf{V}_c)$ are variations in the severity of shift approximating the effect of applying transformations. To evaluate accuracy, the selected classifier f can be unique from the classifier we wish to adapt. This is motivated by the model collinearity phenomenon (Mania and Sra, 2020), in which the relative accuracy on different distributions of data is often the same across multiple classifiers.

6. Dimensionality Reduction

The empirical estimate of the Wasserstein distance converges in sample size to the true Wasserstein distance slowly in large dimensions (Ramdas et al., 2017). Ideally, by reducing the dimensionality of the sample data, fewer samples are needed to achieve an accurate estimate of the Wasserstein distance. We present rigorous theoretical justification that orthonormal projection is a reduction technique that preserves characteristics of the sample data at a distributional level.

Let $m, n \in \mathbb{N}$ and $p \in [1, \infty]$. Call $M(\mathbb{R}^n)$ and $M(\mathbb{R}^m)$ the spaces of probability measures on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. Denote by $M^p(\mathbb{R}^n)$ and $M^p(\mathbb{R}^m)$ the spaces of probability measures having finite p-th moment on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively (here $p = \infty$ is interpreted in the limiting sense of essential supremum). For convenience, we consider only probability measures with densities, so that we do not have to check which measure is absolutely continuous to which other measure (Cai and Lim, 2022, Section III).

Suppose $m \leq n$ and consider the Stiefel manifold on $m \times n$ matrices with orthonormal rows.

$$O(m, n) := \{V \in \mathbb{R}^{m \times n} : VV^\top = I_d\}.$$

For any $V \in O(m, n)$ and $b \in \mathbb{R}^m$, let

$$\varphi_{V,b} : \mathbb{R}^n \to \mathbb{R}^m, \quad x \mapsto \varphi_{V,b}(x) := Vx + b,$$

and for any $\mu \in M(\mathbb{R}^n)$, let $\varphi_{V,b}(\mu) := \mu \circ \varphi_{V,b}^{-1}$ be the pushforward measure. This can be seen as a projection of μ onto the smaller dimensional space $\mathbb{R}^m$, and we call it a *Cai-Lim projection*; it is not unique: it depends on the choice of V and b. Recall then the definition of the p-Wasserstein distance between $\mu, \nu \in M^p(\mathbb{R}^n)$: $W_p(\mu, \nu) := \left[\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^{2n}} \|x - y\|_2^p \, \mathrm{d}\gamma(x, y)\right]^{\frac{1}{p}}$, where $\|\cdot\|_2$ denotes the Euclidean norm and $\Gamma(\mu, \nu) := \{\gamma \in M(\mathbb{R}^{2n}) : \pi_1^n(\gamma) = \nu, \pi_2^n(\gamma) = \mu\}$ is the set of couplings between μ and ν, where π_1^n is the projection onto the first n coordinates and π_2^n is the projection onto the last n coordinates. The following is an important result.

Lemma 5 ***(Cai and Lim, 2022, Lemma II.1)*** *Let $m, n \in \mathbb{N}$ and $p \in [1, \infty]$, and assume $m \leq n$. For any $\mu, \nu \in M^p(\mathbb{R}^n)$, any $V \in O(m, n)$, and any $b \in \mathbb{R}^m$, we have that*

$$W_p\left(\varphi_{V,b}(\mu), \varphi_{V,b}(\nu)\right) \leq W_p\left(\mu, \nu\right).$$

This can be interpreted as "losing some information" when performing a Cai-Lim projection: in smaller dimensional spaces, distributions μ and ν seem to be closer than they actually are. This is an inevitable byproduct of any projection operation. Lemma 5 implies the following corollary.

Corollary 6 *Let $m, n \in \mathbb{N}$ and $p \in [1, \infty]$, and assume $m \leq n$. Consider $\mu, \nu, \rho, \zeta \in M^p(\mathbb{R}^n)$, and pick any $V \in O(m, n)$ and any $b \in \mathbb{R}^m$. Suppose $W_p(\mu, \nu) \geq W_p(\rho, \zeta)$. Then, there exists $\varepsilon > 0$ such that if $W_p\left(\mu, \nu\right) - W_p\left(\varphi_{V,b}(\mu), \varphi_{V,b}(\nu)\right) \leq \varepsilon$, then*

$$W_p\left(\varphi_{V,b}(\mu), \varphi_{V,b}(\nu)\right) \geq W_p\left(\varphi_{V,b}(\rho), \varphi_{V,b}(\zeta)\right).$$

Proof Set $\varepsilon = W_p(\rho, \zeta) - W_p(\varphi_{V,b}(\rho), \varphi_{V,b}(\zeta))$. The result follows immediately by Lemma 5. ■

Corollary 6 states the following. If we "do not lose too much information" when performing a Cai-Lim projection of the two farthest apart distributions, then the inequality $W_p(\mu, \nu) \geq W_p(\rho, \zeta)$ between the original distribution is preserved between their projections. A more intuitive discussion, as well as empirical justification, can be found in Appendices B and C of Lin et al. (2023).

7. Related Work

Distribution shifts can make deep learning models act dangerously in the real world (Narasimhamurthy et al., 2019). Some offline techniques aim to improve classification robustness to distribution shift at training time. These include data augmentation, which expands the experiences that a model would be exposed to during training (Hendrycks et al., 2019; Erichson et al., 2022; Verma et al., 2019; Yun et al., 2019; Kim et al., 2020b; Hendrycks et al., 2020). Another offline approach is domain adaptation, in which models are adapted to perform well on unlabeled data in some new distribution (Ben-David et al., 2010; Bousmalis et al., 2017; Hoffman et al., 2018). Offline approaches lack flexibility to shifts unforeseen at train time, but they can be combined with online methods like `SuperStAR` to further improve performance.

Alternate techniques aim to handle distribution shift online. For example, test-time adaptation methods perform additional training at inference time in response to incoming data (Gandelsman et al., 2022; Wang et al., 2022; Mummadi et al., 2021). Test time augmentation approaches instead apply transformations to the input, then ensemble predictions if multiple transformations are applied (Simonyan and Zisserman, 2014; Krizhevsky et al., 2012; He et al., 2016; Guo et al., 2017; Mummadi et al., 2021; Kim et al., 2020a; Lyzhov et al., 2020). In contrast, `SuperStAR` selects transformations online without querying the model. Furthermore, our work uniquely identifies MDPs as an equivalent formulation of the transformation selection problem, enabling the application of standard RL techniques. We further explore related work in Appendix D of Lin et al. (2023).

8. Application: ImageNet-C

The ImageNet-C dataset (Hendrycks and Dietterich, 2019) is constructed from ImageNet samples corrupted by 19 semantic preserving transformations. We deploy the learned policy π in `SuperStAR` to correct ImageNet-C corruptions, and we evaluate the accuracies of Resnet-50 classifiers with and without correction. We evaluate a baseline classifier trained without data augmentation and classifiers trained with data augmentation through AugMix, NoisyMix, DeepAugment, DeepAugment with Augmix, and Puzzlemix. We also evaluate the ability to generalize outside of the ImageNet-C benchmark. Experiment details can be found in Appendix E of Lin et al. (2023).[3]

ImageNet-C. Table 1 summarizes the classifier accuracy improvements from applying `SuperStAR` to ImageNet-C. For brevity, we exclude from Table 1 shifts for which `SuperStAR` incurs no change in accuracy (see Appendix F of Lin et al. (2023) for full table). Corrections via our `SuperStAR` algorithm lead to accuracy improvements for a majority of shifts, with maximum improvement of 14.21% (averaged across all five severity levels). In general accuracy improvements are greater for higher severities (due to space constraints, per-severity accuracy improvements are shown in Appendix F of Lin et al. (2023)). Furthermore, when combined with data augmentation, `SuperStAR`

3. Our code can be found at https://github.com/vwlin/SuperStAR.

Table 1: Average accuracies (%) on each ImageNet-C shift with and without `SuperStAR` for ResNet-50 classifiers. Accuracy improvement is denoted by Δ = R (recovered) $-$ S (shifted). Values are over 5 severity levels with 3 trials each. "gaussian" refers to Gaussian noise.

	No Data Aug			AugMix			NoisyMix			DeepAugment			DeepAug+AugMix			PuzzleMix		
shift	S	R	Δ	S	R	Δ	S	R	Δ	S	R	Δ	S	R	Δ	S	R	Δ
none	74.52	74.52	0.00	75.94	75.94	0.00	76.22	76.22	0.00	75.86	75.86	0.00	75.26	75.26	0.00	75.63	75.63	0.00
gaussian	31.11	43.44	12.33	41.90	50.87	8.98	52.71	55.51	2.80	59.07	59.48	0.41	55.39	61.43	6.05	41.48	46.94	5.46
shot	28.61	42.81	14.21	41.78	50.93	9.15	51.81	55.38	3.57	58.21	58.46	1.24	55.76	62.37	6.61	37.39	45.56	7.77
impulse	26.57	39.36	12.79	38.78	47.49	8.71	50.73	53.37	2.64	58.61	58.38	-0.23	55.16	60.67	5.50	35.28	42.82	7.54
snow	30.51	29.28	-1.13	37.89	36.85	-1.04	43.20	41.52	-1.68	41.71	39.91	-1.80	47.68	46.36	-1.32	39.48	37.74	-1.75
frost	35.16	35.07	-0.09	41.39	41.24	-0.15	50.05	49.46	-0.59	46.87	46.08	-0.78	51.21	50.26	-0.95	46.96	45.75	-1.21
brightness	65.17	65.72	0.55	67.35	68.46	1.11	68.82	69.87	1.05	69.04	69.73	0.69	69.42	70.18	0.76	69.59	69.67	0.08
contrast	35.56	37.69	2.14	48.96	49.85	0.89	50.37	52.74	2.37	44.89	48.23	3.33	56.01	57.40	1.39	50.56	52.87	2.30
speckle	36.09	49.26	13.18	50.61	56.94	6.33	57.67	60.88	3.22	62.21	63.81	1.59	60.93	65.66	4.74	42.24	51.92	9.68
spatter	46.65	46.44	-0.20	53.25	52.93	-0.32	57.63	57.34	-0.29	53.74	53.58	-0.16	57.75	57.61	-0.14	53.27	52.95	-0.32
saturate	59.00	59.17	0.17	61.42	61.89	0.47	63.48	64.00	0.52	64.59	64.81	0.22	65.79	66.12	0.33	65.96	65.60	-0.37

often leads to higher accuracies than `SuperStAR` or data augmentation alone. In the cases of no shift and the shifts not shown in Table 1, `SuperStAR` refrains from taking any action and does not affect accuracy. This is examined further in Appendix G of Lin et al. (2023).

Interestingly, for a hyperparameter selection that allows some increase in Wasserstein distance at each step ($\beta = 1.12$), we find that `SuperStAR` selects a non-trivial 5-action sequence of transformations for contrast shift severity level 5, which incrementally increases the accuracy of the AugMix classifier. Figure 2 shows a sample image with the applied transformations and resulting accuracies. The transformations incur a noticeable change in the image.

Generalization beyond ImageNet-C. We also evaluate the ability of `SuperStAR` to generalize outside of the ImageNet-C benchmark. 1) We construct composite ImageNet-C shifts from pairs of the ImageNet-C corruptions for which `SuperStAR` improves accuracy. To each shift, we reapply our operability classifier and policy network without retraining. Table 2 shows the average classifier accuracy improvements from `SuperStAR`. For nearly all shifts, our method improves classifier accuracy, with a maximum improvement in average accuracy of 9.81% (see Appendix H of Lin et al. (2023) for severity level breakdowns). 2) We also apply our method to CIFAR-100-C (Hendrycks and Dietterich, 2019), an analagous benchmark to ImageNet-C. We use the pretrained policy network and retrain only the operability classifier on the surrogate corruptions regenerated for CIFAR-100. We evaluate on a variety of Wide ResNets trained with data augmentation (AugMix, NoisyMix, and PuzzleMix) and without (baseline).[4] Appendix I of Lin et al. (2023) contains further details. Table 3 shows the accuracy improvements from applying `SuperStAR` to CIFAR-100-C. Without any retraining of the policy network, `SuperStAR` improves average accuracy by up to 8.25% (see Appendix J of Lin et al. (2023) for full table and severity level breakdowns). In some cases, such as impulse noise, our method decreases accuracy for the classifiers trained with data augmentation, while increasing that of the baseline classifier. We attribute this to the operability classifier mislabeling such shifts as operable.

Overall, `SuperStAR` demonstrates an ability to dynamically respond to distribution shift online, selecting context-appropriate actions (or inaction) and significantly improving classification accuracy on ImageNet-C for a variety of corruptions. In some cases, `SuperStAR` identifies complex sequences of corrective transformations. We attribute this strong performance largely to our selection of surrogate shifts (see Appendix K of Lin et al. (2023) for more discussion). `SuperStAR`

4. DeepAugment and DeepAugment with Augmix are not evaluated on CIFAR-100 in the original publications.

Table 2: Average accuracies (%) on each composite ImageNet-C shift with and without SuperStAR for ResNet-50 classifiers. Composites are combinations of Gaussian noise (GN), shot noise (ShN), impulse noise (IN), speckle noise (SpN) with brightness (B), contrast (C), saturate (S). Accuracy improvement is denoted Δ = R (recovered) − S (shifted). Values are over 5 severity levels with 3 trials each.

	No Data Aug			AugMix			NoisyMix			DeepAugment			DeepAug+AugMix			PuzzleMix		
shift	S	R	Δ	S	R	Δ	S	R	Δ	S	R	Δ	S	R	Δ	S	R	Δ
GN + B	23.75	30.65	6.90	29.03	37.20	8.17	42.87	44.37	1.50	51.29	49.80	-1.49	43.03	52.17	9.11	33.25	35.15	1.90
GN + C	22.39	22.39	0.00	29.40	29.40	0.00	35.96	35.96	0.00	34.00	34.00	0.00	41.31	41.31	0.00	32.46	32.46	0.00
GN + S	19.35	26.37	7.02	26.65	33.43	6.79	37.13	38.50	1.37	42.83	44.50	1.68	42.41	50.82	8.41	28.96	30.98	2.02
IN + B	22.03	30.08	8.04	29.62	37.91	8.29	43.66	45.89	2.24	52.47	52.90	0.43	44.14	53.95	9.81	32.13	35.65	3.52
IN + C	18.41	18.41	0.00	25.89	25.89	0.00	33.05	33.05	0.00	31.61	31.61	0.00	39.72	39.72	0.00	26.27	26.27	0.00
IN + S	17.59	24.61	7.02	25.24	30.37	5.12	34.23	36.07	1.84	40.96	42.34	1.37	43.58	49.05	5.47	25.74	29.37	3.62
ShN + B	22.32	30.08	7.76	28.38	35.82	7.44	40.26	41.64	1.38	49.44	47.65	-1.79	41.38	50.61	9.23	31.84	33.63	1.79
ShN + C	21.13	21.03	-0.10	28.36	27.94	-0.42	35.22	34.98	-0.24	34.26	33.96	-0.30	41.05	40.76	-0.29	30.13	30.26	**0.13**
ShN + S	18.49	26.03	7.54	27.33	33.74	6.40	37.58	39.11	1.52	42.47	44.84	2.37	44.04	51.26	7.22	26.33	30.03	3.70
SpN + B	27.20	32.43	5.23	32.64	38.55	5.91	43.60	45.64	2.05	52.77	52.37	-0.41	45.82	52.99	7.17	35.80	37.81	2.01
SpN + C	24.00	24.18	0.18	32.70	32.58	-0.12	39.21	39.57	0.36	36.30	36.98	0.68	44.19	44.49	0.30	33.40	33.74	0.34
SpN + C	23.80	31.36	7.56	35.87	40.22	4.35	45.04	45.98	0.93	47.64	50.87	3.23	50.25	55.98	5.72	30.55	35.94	5.39

Table 3: Average accuracies (%) on each CIFAR-100-C shift with and without SuperStAR for Wide ResNet classifiers. Accuracy improvement is denoted by Δ = R (recovered) − S (shifted). Values are over 5 severity levels with 3 trials each.

	No Data Aug			AugMix			NoisyMix			PuzzleMix		
shift	S	R	Δ	S	R	Δ	S	R	Δ	S	R	Δ
none	81.13	81.13	0.00	76.28	76.28	0.00	81.29	81.29	0.00	84.01	84.01	0.00
gaussian noise	21.12	26.81	5.70	47.89	51.07	3.18	65.91	66.34	0.43	20.87	28.18	7.31
shot noise	29.96	36.34	6.38	55.69	58.24	2.55	70.39	70.72	0.33	31.12	39.37	8.25
impulse noise	19.21	26.09	6.88	59.68	59.09	-0.59	79.72	76.08	-3.64	37.18	37.01	-0.17
glass blur	20.68	26.61	5.93	54.08	56.09	2.00	58.82	60.48	1.66	31.07	37.62	6.55
speckle noise	31.58	35.76	4.18	58.11	59.33	1.23	71.67	71.57	-0.09	33.96	38.94	4.98
spatter	61.23	62.23	1.00	72.28	71.25	-1.02	78.11	77.44	-0.67	79.73	78.90	-0.83
saturate	68.82	68.88	0.06	64.52	64.77	0.25	69.83	70.21	0.39	72.93	72.99	0.05

also generalizes to distribution shifts outside of the ImageNet-C benchmark, without retraining the policy network. Finally, we note that although promising, SuperStAR is limited in its speed of response and its reliance on appropriately selected actions and surrogate corruptions. We further discuss these limitations and more in Appendix L of Lin et al. (2023).

9. Conclusion

In this work we presented SuperStAR, which uses control for learned models to detect and recover from distribution shift. SuperStAR uses the Wasserstein distance (with a theoretically-sound approach for dimensionality reduction using orthonormal projections) to detect distribution shifts and select recovery actions from a library of image correction techniques. We formulate this action selection problem as a Markov decision process, and we train the policy for computing actions using reinforcement learning. To hedge against harmful actions, we employ a binary classifier to check a minimum condition for our method to operate on corrupted data. We applied our approach to various classifiers on the ImageNet-C dataset, and we obtained significant accuracy improvements when compared to the classifiers alone. Finally, we showed that SuperStAR generalizes to composite ImageNet-C shifts and CIFAR-100-C with no retraining of the policy. Expansion of the action library and additional tuning can lead to further improvements on these benchmarks.

Acknowledgments

This research was supported in part by ARO W911NF-20-1-0080. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government.

References

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*, pages 1–12, 2011.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

Yuhang Cai and Lek-Heng Lim. Distances between probability distributions of different dimensions. *IEEE Transactions on Information Theory*, 2022.

N. Benjamin Erichson, Soon Hoe Lim, Francisco Utrera, Winnie Xu, Ziang Cao, and Michael W. Mahoney. Noisymix: Boosting robustness by combining data augmentations, stability training, and noise injections. *CoRR*, abs/2202.01263, 2022. URL https://arxiv.org/abs/2202.01263.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks, October 2018. URL http://arxiv.org/abs/1610.02136. arXiv:1610.02136 [cs].

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *CoRR*, abs/2006.16241, 2020. URL https://arxiv.org/abs/2006.16241.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018.

Ildoo Kim, Younghoon Kim, and Sungwoong Kim. Learning loss for test-time augmentation. *Advances in Neural Information Processing Systems*, 33:4163–4174, 2020a.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. *CoRR*, abs/2009.06962, 2020b. URL https://arxiv.org/abs/2009.06962.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Vivian Lin, Kuk Jin Jang, Souradeep Dutta, Michele Caprio, Oleg Sokolsky, and Insup Lee. Dc4l: Distribution shift recovery via data-driven control for deep learning models. *arXiv preprint arXiv:2302.10341*, 2023.

Alexander Lyzhov, Yuliya Molchanova, Arsenii Ashukha, Dmitry Molchanov, and Dmitry Vetrov. Greedy policy search: A simple baseline for learnable test-time augmentation. In *Conference on Uncertainty in Artificial Intelligence*, pages 1308–1317. PMLR, 2020.

Horia Mania and Suvrit Sra. Why do classifier accuracies show linear trends under distribution shift? *arXiv preprint arXiv:2012.15483*, 2020.

Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*, 2021.

Monal Narasimhamurthy, Taisa Kushner, Souradeep Dutta, and Sriram Sankaranarayanan. Verifying conformance of neural network models: Invited paper. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019. doi: 10.1109/ICCAD45719.2019.8942151.

Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. doi: 10.1109/TIP.2003.819861.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019. URL http://arxiv.org/abs/1905.04899.