This CVPR 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Limitations of Post-Hoc Feature Alignment for Robustness

Collin Burns
UC Berkeley

Jacob Steinhardt
UC Berkeley

Abstract

Feature alignment is an approach to improving robustness to distribution shift that matches the distribution of feature activations between the training distribution and test distribution. A particularly simple but effective approach to feature alignment involves aligning the batch normalization statistics between the two distributions in a trained neural network. This technique has received renewed interest lately because of its impressive performance on robustness benchmarks. However, when and why this method works is not well understood. We investigate the approach in more detail and identify several limitations. We show that it only significantly helps with a narrow set of distribution shifts and we identify several settings in which it even degrades performance. We also explain why these limitations arise by pinpointing why this approach can be so effective in the first place. Our findings call into question the utility of this approach and Unsupervised Domain Adaptation more broadly for improving robustness in practice.

1. Introduction

A foundational assumption made in most of machine learning is that the training distribution is identical to the test distribution. However, this assumption is commonly violated in practice, which can substantially decrease the performance of models [10, 24]. This can be especially problematic in high-stakes applications such as autonomous vehicles. One way of improving robustness is to exploit unlabeled test data to adapt the model to the new distribution. This process is called Unsupervised Domain Adaptation (UDA) [32].

A common approach in UDA, known as feature alignment or domain alignment, is to align the feature activations between the source and target distributions [5–8, 13, 17, 20, 26, 29, 30, 32]. Feature alignment has also been applied beyond UDA in domains such as causal inference [15, 28]. Simple forms of feature alignment normalize the features of a trained model so that the training set and test set have the same first and second order statistics in some feature space [19, 30], while other approaches match distributions in more complicated ways, such as by being indistinguishable to an adversarial discriminator [8, 20].

We focus on one simple feature alignment method: Adaptive Batch Normalization (AdaBN) [19]. Like many other popular and effective feature alignment methods (e.g. Cariucci et al. [5], Roy et al. [26], Sun and Saenko [31], Sunet al. [30], Wang et al. [33]), AdaBN is normalization-based, meaning it matches first and second order statistics between the two feature distributions. It is also a post-hoc method, meaning it aligns features for a model that has already been trained, making it particularly simple and applicable even for unforseen distribution shifts. Given a neural network trained on source data with Batch Normalization (BN) [14], AdaBN re-estimates the BN statistics of that model using the target data. In other words, AdaBN aligns the mean and variance of each channel in the network across the two distributions.

Despite its simplicity, in recent work Nado et al. [21], Schneider et al. [27] showed that aligning batch norm statistics between the train and test distributions can be used to achieve state-of-the-art accuracy on the robustness benchmark ImageNet-C [10]. Schneider et al. [27] argue that we should therefore start using normalization-based feature alignment methods whenever we evaluate robustness. Nado et al. [21] additionally find that aligning BN statistics does not help as much for some other types of distribution shift. However, neither paper describes why this method works well on ImageNet-C or why it does not help as much with other types of distribution shift.

We build on this work by investigating when and why methods like AdaBN help. Our findings include:

- Showing that aligning BN statistics can actually *degrade* accuracy on several types of distribution shift, both conceptually and in practice.
- Identifying implicit symmetry assumptions made by these methods and showing how violations of these assumptions can cause performance degradation.
- Demonstrating and explaining how aligning BN statistics primarily helps with distribution shifts that involve changes in local image statistics.

Our findings have several implications. While aligning BN statistics is an effective method for improving robustness

in some settings, it only significantly helps on a narrow set of distribution shifts and can even degrade performance. These limitations may prevent it from being useful in practical applications. Furthermore, we find that existing justifications of feature alignment are inadequate for explaining when and why these methods work. Future work on UDA should explicitly identify the properties of data distributions and neural networks that these methods rely on in practice. Finally, some of our findings apply to UDA more broadly, calling into question whether UDA is a strong approach to improving the robustness of machine learning systems in the first place. More work is therefore needed to make UDA practical for improving robustness.

2. Related Work

We focus on feature alignment methods that work by aligning the Batch Normalization statistics between the source and target distributions for a trained neural network. In this section, we describe how this relates to other feature alignment methods, and we describe why existing justifications for feature alignment do not adequately explain their practical success.

Feature Alignment Methods. Several UDA methods closely resemble AdaBN by similarly aligning normalization statistics of trained models. Sun et al. [30] whiten and re-color the target distribution to match the mean and covariance of the source distribution in the input. Sun and Saenko [31] extend this by matching the mean and covariance in a neural network layer, rather than in the input. Because these are post-hoc methods based on normalization like AdaBN, our findings directly apply to them as well.

Some UDA methods are normalization-based but require modifying the *training* of neural networks as well. Cariucci et al. [5] modify AdaBN by learning a linear combination of source and target Batch Normalization statistics. Wang et al. [33] introduce a new layer for UDA that uses domain-specific Batch Normalization statistics and that automatically adapts to the transferability of different channels. Some, but not all, of our findings apply to these methods as well.

In a related vein, adversarial alignment methods such as Ganin et al. [8], Long et al. [20] learn feature representations for which a discriminator cannot distinguish source and target data. Unlike the normalization-based approaches that we focus on in this work, adversarial methods aim to learn feature representations that are completely indistinguishable instead of only matching first and second order statistics, and again modify the training of networks, which can be expensive. These methods can improve performance, but they are also much less efficient than post-hoc feature alignment methods.

Justifications of Feature Alignment are Inadequate. Many papers that introduce feature alignment methods intuitively suggest that matching feature distributions makes the features more domain-invariant and consequently mitigates the effects of distribution shift [5, 7, 30, 31]. However, aligning the features between two distributions is not sufficient for good test performance in general because aligning the marginal distributions $p_S(x)$ and $p_T(x)$ in some feature space may not align the *class-conditional* distributions $p_S(x|y)$ and $p_T(x|y)$ [16, 35].

Some papers (e.g. Ganin et al. [8], Long et al. [20]) motivate aligning feature distributions by referring to David et al. [2], which introduces generalization bounds for UDA. For a given hypothesis class $\mathcal{H}$ and feature space, these bounds guarantee good test performance as long as (i) the two distributions are "indistinguishable" with respect to $\mathcal{H}$, and (ii) there is a hypothesis $h \in \mathcal{H}$ that simultaneously does well on both distributions. However, Johansson et al. [16], Zhao et al. [35] recently described problems with this theory, and in the Supplementary Material we argue that these generalization bounds are probably vacuous in practice. In contrast to this work, we focus on empirically understanding when and why aligning BN statistics works *in practice*.

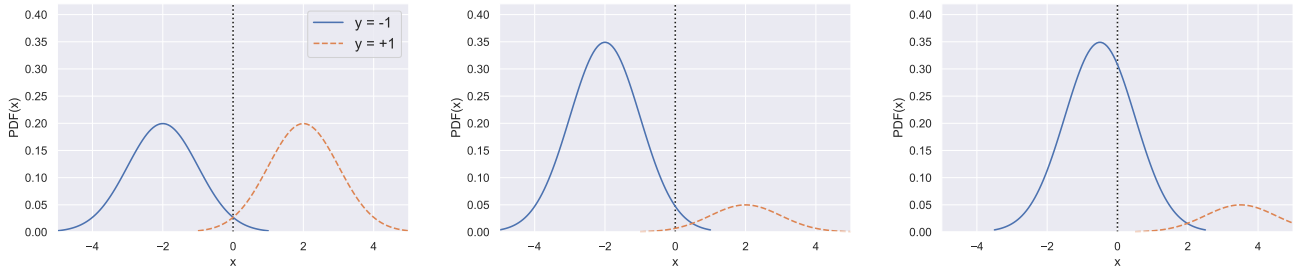
Several impossibility theorems show that successful UDA requires strong assumptions on the source and target distributions [3, 4]. Nevertheless, many feature alignment methods are effective in practice. This raises the question: *What properties of distribution shifts and neural networks does feature alignment exploit to improve robustness?* We answer this question for AdaBN in the process of investigating its limitations.

3. Failure Modes of AdaBN

In this section, we characterize when normalization-based methods *hurt* accuracy. Prior work showed that feature alignment can degrade performance under label shift, i.e. $p_S(y) \neq p_T(y)$ [16, 25, 35]. We extend these earlier observations by showing that label shift also has a more severe impact on *deep* layers than on *shallow* layers.

We then construct two additional failure modes that can occur even when the label distribution doesn't change, i.e. $p_S(y) = p_T(y)$, and even under the covariate shift assumption, i.e. when $p_S(y|x) = p_T(y|x)$. In particular, we show that normalization-based alignment methods can fail when either different *examples* or *spatial locations* are shifted in qualitatively different ways, and we show that both types of shift can arise in practice. This suggests that these methods would be unreliable in safety-critical applications involving unforeseen distribution shifts.

For each of the three failure modes we exhibit, we first provide a simple conceptual example of why the failure mode is possible, then demonstrate the failure on real data.



(a) Source distribution. (b) Target distribution. (c) Mean-normalized target distribution.

Figure 1: An illustration of how aligning the means between the original and shifted distributions can hurt accuracy when there is label shift. Aligning the variances does not change the accuracy in this case. The blue curve is the PDF for $y = -1$ and the orange (dashed) curve is the PDF for $y = +1$. The dashed line indicates the decision boundary of the classifier. Both classes are initially equally likely (Figure 1a). After shifting the label distribution, the accuracy of the original classifier remains high (Figure 1b), but decreases after normalizing the mean (Figure 1c).

Table 1: Accuracy of each method on CIFAR-10 (C-10), TinyImageNet (TIN), ImageNet (IN), CIFAR-10-C (C-10-C), TinyImageNet-C (TIN-C), ImageNet-C (IN-C), ImageNetV2 (INV2), and Stylized ImageNet (SIN).

METHOD	C-10	TIN	IN	C-10-C	TIN-C	IN-C	INV2	SIN
ORIGINAL MODEL	94.8	63.8	76.1	72.3	24.7	38.1	63.2	7.1
ADABN	92.8	60.3	75.6	83.6	40.1	46.9	60.9	10.2

3.1. Experimental Setup

We begin by describing the experimental setup that we use for the remainder of the paper. Code for the experiments is available at https://github.com/collin-burns/feature-alignment.

Datasets. We evaluate models on a diverse set of distribution shift datasets. In several experiments we use the robustness benchmarks CIFAR-10-C, TinyImageNet-C, and ImageNet-C [10]. These datasets include 15 noise, blur, weather, and digital corruptions with 5 severities for each; we apply AdaBN on each corruption and severity independently then average the resulting accuracies. We also run experiments on ImageNetV2 [24] and Stylized ImageNet [9]. ImageNetV2 was constructed by trying to reproduce how the ImageNet dataset was collected. This distribution shift reduces the accuracy of our ImageNet-trained model from 76% to 63%. Stylized ImageNet [9] changes the texture and style of ImageNet images in a variety of ways. This more severe shift reduces the accuracy all the way down to 10%. As we are working in the context of domain adaptation, for each dataset we assume we have access to all of the unlabeled target data.

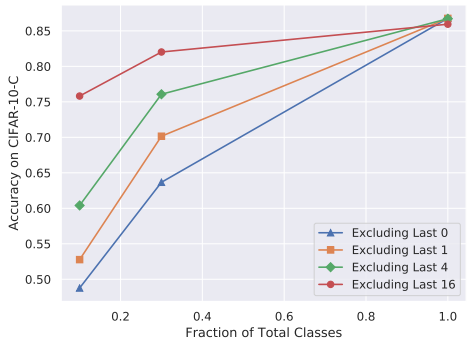
Using these datasets has two main advantages. First, ImageNet-C, ImageNetV2, and Stylized ImageNet are qualitatively distinct shifts, but they are readily comparable because they are all based on ImageNet. Second, the corruption datasets make it possible to compare different severities of shift while controlling for other factors, which we make use of in some experiments.

Models. We use the pre-trained ResNet-50 model included in the `torchvision` package [23] for all ImageNet experiments. For all CIFAR-10 and TinyImageNet experiments, we use a 40-2 WideResNet [34] trained for 100 epochs using SGD with momentum 0.9, initial learning rate 0.1, weight decay 0.0005, dropout rate 0.3, and batch size 128. For data augmentation, we use random cropping with zero-padding 4 and random horizontal flips. We show the performance of these models on each dataset in Table 1.

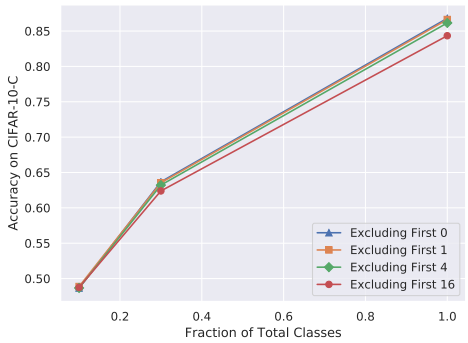
3.2. Shifted Label Distribution

We now show how it can hurt to normalize the feature distributions when $p_S(y) \neq p_T(y)$. Similar observations were made in prior work [16, 25, 35], but we extend this by investigating how it occurs in more detail and by also showing that deeper layers are more sensitive to label shift than shallow layers.

Conceptual Example. Consider binary classification when y is sampled uniformly from $\{-1, 1\}$, and $x \sim \mathcal{N}(2y, 1)$. The Bayes classifier is $f(x) = \text{sign}(x)$. Suppose we shift the label distribution so that $p(y = -1) =$



(a) CIFAR-10-C Excluding Last Layers



(b) CIFAR-10-C Excluding First Layers

Figure 2: The effect of updating the Batch Normalization statistics (AdaBN) on CIFAR-10-C in all but the last k BN layers (left) and all but the first k BN layers (right) as a function of the number of classes kept in the target shift, for $k \in \{0, 1, 4, 16\}$. There are 37 BN layers in total. AdaBN does worse as the number of classes decreases. We can mitigate this decrease by excluding some of the final Batch Normalization layers from feature alignment, but not by excluding some of the first Batch Normalization layers. This indicates that deep layers are more sensitive to label shift than shallow layers.

$\frac{7}{8}$ for the target distribution. The new mean is then $-\frac{3}{2}$. If we normalize the mean to match the original mean of zero, this pushes the $y = -1$ mode from $\mathcal{N}(-2, 1)$ to $\mathcal{N}(-\frac{1}{2}, 1)$ and increases the classification error substantially, as illustrated in Figure 1. This is despite the fact that the classifier would have had high accuracy without any normalization.

In Practice. We now exhibit this issue on CIFAR-10. From the discussion above, we should expect the accuracy of AdaBN to degrade as it is applied to a smaller fraction of classes. We confirm this and show the results in Figure 2 (blue curve). Specifically, we evaluate the accuracy of AdaBN applied to subsets of CIFAR-10-C classes, while still allowing the classifier to output any of the 10 classes. Making some classes occur with probability zero is an extreme form of class reweighting, but one that could still arise in practice. For simplicity, we use the first k classes for different values of k. In the worst case of a single class, the accuracy falls below 50%.

Shallow vs Deep Layers. Intuitively, shallow layers capture low-level information like edges and colors, which should be mostly class-agnostic, while deeper layers capture more abstract, class-specific representations. This suggests that only updating the Batch Normalization statistics in the earlier layers may mitigate the drop in accuracy caused by applying AdaBN under label shift.

We confirm this and show the results in Figure 2a. When one doesn't update the last 16 (out of 37) Batch Normalization layers, accuracy remains high even when AdaBN is applied to a single class. To check that this is due to excluding the *final* layers, we also test not updating the first k Batch Normalization layers and confirm that it does not improve performance (Figure 2b). We find similar results on other datasets; see the Supplementary Material for details.

3.3. Shifted Spatial Locations

A second type of failure mode occurs when there are different shifts for different spatial locations. This can occur if, for example, a border is added to every image, as this results in the distribution of boundary pixels changing dramatically without the distribution of interior pixels changing at all. We illustrate this in Figure 3 (top row). Unlike the previous failure mode, this can arise even under the covariate shift assumption and when the class distribution is fixed.

Conceptual Example. Again consider binary classification when y is sampled uniformly from $\{-1, +1\}$. Let $\mathbf{x} = (x_1, x_2)$, where $x_1 \sim \mathcal{N}(4 + 2y, 1)$ and $x_2 \sim \mathcal{N}(4, 1)$. The classifier $f(\mathbf{x}) = \text{sign}(x_1 - 4)$ has high accuracy.

We use the features x_1 and x_2 to model different spatial dimensions in a convolutional channel. Since Batch Normalization computes the mean and variance over both a batch of examples and all spatial locations within a channel, we simultaneously normalize $\mathbf{x}$ over both samples and coordinates. Specifically, AdaBN matches the mean and variance of a shifted input $\mathbf{x}$ by transforming the input to be

$$\tilde{\mathbf{x}} = \frac{\sigma_s}{\sigma_t}(\mathbf{x} - \mu_t \cdot \mathbf{1}) + \mu_s \cdot \mathbf{1}\,, \tag{1}$$

where $\mathbf{1} := (1, 1) \in \mathbb{R}^2$. Imagine we shift the distribution by making $x_2 = 0$ for each example. This doesn't change the accuracy of f, but it decreases the mean and variance of $\mathbf{x}$. The original mean and variance were the average mean and variance over each dimension: $\mu_s = \frac{1}{2}(\mathbb{E}[x_1] + \mathbb{E}[x_2]) = 4$ and $\sigma_s^2 = \frac{1}{2}(\text{var}(x_1) + \text{var}(x_2)) = 1$. Under the shift,

$\mathbb{E}[x_2] = \text{var}(x_2) = 0$, so the mean and variance become $\mu_t = \frac{1}{2}(\mathbb{E}[x_1] + 0) = 2$ and $\sigma_t^2 = \frac{1}{2}(\text{var}(x_1) + 0) = \frac{1}{2}$.

For the values given above, we have $\tilde{x}_1 = \sqrt{2}(x_1 - 2) + 4$. The mode corresponding to $y = -1$ is initially centered at $x_1 = 2$, so after normalizing it shifts to $\tilde{x}_1 = 4$. Since the decision boundary of f passes through $x_1 = 4$, the new error of f conditioned on $y = -1$ is $\frac{1}{2}$, which is higher than it was before applying AdaBN.

Table 2: Accuracy on the Black Border distribution shift.

METHOD	C-10	TIN
ORIGINAL MODEL	65.0	22.6
ADABN	52.5	11.8

In Practice. We now exhibit an analogous failure mode on real data. Our example uses the "black border" transformation, where we remove all boundary pixels by replacing them with zero. Similar to the conceptual example, this shifts the distribution of some spatial locations but not others. We evaluate the robustness of models to this transformation on CIFAR-10 and TinyImageNet, where we chose the width of the border to be $1/4$ the length of the image, so that 25% of the area of the image remains.

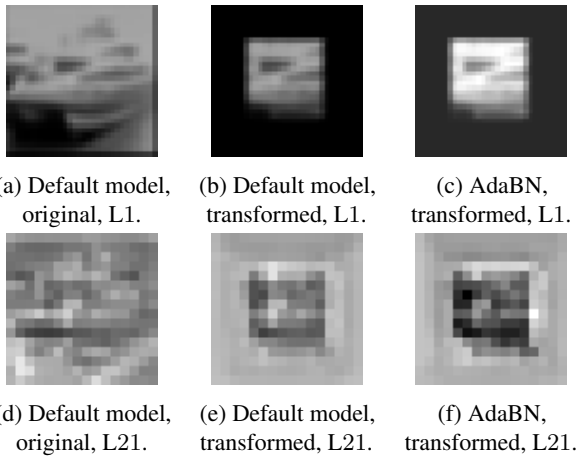
The results are given in Table 2. Applying AdaBN to this transformation hurts accuracy relative to the original model, almost cutting it in half for TinyImageNet.

To verify that the drop in performance comes from shifted spatial locations, we visualize the effect of Batch Normalization on the activations of the model. We show representative channel activations after the first and twenty-first Batch Normalization layers in Figure 3. For both layers, AdaBN changes the scale of the activations so that the mean is closer to that of a typical in-distribution activation. Since the border pixels are either darker than a normal input (top) or brighter than a normal input (bottom), matching the mean throws off the scale of the center of the image, which contains the actual content.

3.4. Shifted Examples

Finally, we show that normalization-based feature alignment can fail if different examples are subject to different shifts. One example of this is with label shift, but we show that it is a more general phenomenon that can occur even when $p_S(y) = p_T(y)$. Specifically, it can naturally occur when the distribution of a single spatial location is multimodal, such as for a mixture distribution.

Conceptual Example. Consider binary classification again, but this time suppose $x|y$ is a mixture of Gaussians. For simplicity, we focus on $y = -1$ and assume that we normalize separately for each class. Define the source distribution by drawing $x|(y = -1)$ from the mixture distribution $\frac{1}{2}N(-9, 1) + \frac{1}{2}N(-1, 1)$. The classifier $f(x) = \text{sign}(x)$ initially has low error conditioned on $y = -1$. Suppose we now reweight the modes of $x|(y = -1)$ so that we instead sample from $\frac{3}{4}N(-9, 1) + \frac{1}{4}N(-1, 1)$. This decreases both the variance and the mean. Normalizing to have the original mean and variance then pushes the $N(-1, 1)$ mode to be greater than 0, resulting in a larger classification error. We illustrate this in Figure 4.



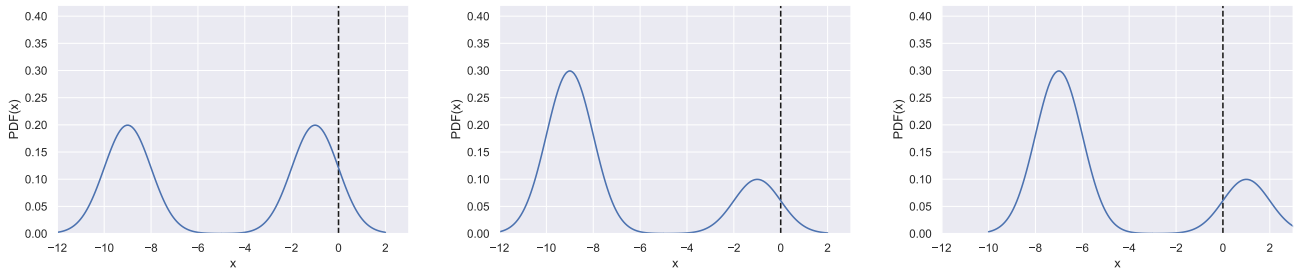
(a) Default model, original, L1. (b) Default model, transformed, L1. (c) AdaBN, transformed, L1.

(d) Default model, original, L21. (e) Default model, transformed, L21. (f) AdaBN, transformed, L21.

Figure 3: A representative channel in the 1st Batch Normalization layer (top row, L1) and 21st Batch Normalization layer (bottom row, L21). The scales of the images (the minimum and maximum values, corresponding to black and white) are the same. Updating the Batch Normalization statistics (Figures 3c and 3f) changes the magnitude of the activations even though the original activations are more appropriate (Figures 3a and 3d).

In Practice. In the conceptual example described above, individual coordinates for a single class were distributed according to a mixture distribution. To exhibit an analogous shift on real data, we first identify a surprising phenomenon. We find that applying AdaBN to the original test data (or even ImageNetV2) can degrade accuracy by a few percentage points (see Table 1). This is because models use data augmentation during training but not at test time, which can lead to a discrepancy if one naively aligns the train and test sets while only applying augmentation to the former.

We find that one can prevent this decrease in accuracy by updating the Batch Normalization statistics on the target data while using the augmentation used during training time, which mimics how the original Batch Normalization statistics were computed. We denote this approach by "AdaBN + Aug." We only use augmentations for aligning the features but do not use them at test time. Table 3 displays the accura-



(a) Source distribution. (b) Target distribution. (c) Mean-aligned target distribution.

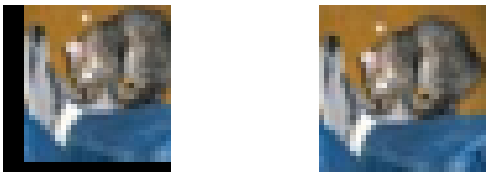
Figure 4: An illustration of how aligning the class-specific means can hurt accuracy when the class feature distribution is multi-modal. The blue curve is the PDF of $x|y=-1$ and has two modes. For clarity, we do not show the PDF of $x|y=+1$. The dashed line indicates the decision boundary of the classifier. After shifting the feature distribution so that the mode centered at -9 becomes more likely, the accuracy remains high. However, aligning the source and target means pushes the mode centered at -1 to be greater than 0, causing a drop in accuracy.

Table 3: Accuracy (in percent) of AdaBN with training augmentations (+ Aug) on each dataset. The change in accuracy from AdaBN is given in parentheses. AdaBN + Aug does better in most cases, especially on CIFAR-10 and TinyImageNet.

METHOD	C-10	TIN	IN	C-10-C	TIN-C	IN-C	INV2	SIN
ADABN + AUG	94.8 (+2.0)	64.0 (+3.7)	76.0 (+0.4)	86.7 (+3.1)	41.8 (+1.7)	43.3 (-3.6)	63.8 (+2.9)	8.4 (-1.8)

cies for AdaBN + Aug on each dataset. This modification consistently improves in-distribution accuracy and often improves OOD accuracy. For example, it improves accuracy on ImageNetV2 from 60.9% to 63.8%, compared to 63.2% for standard AdaBN, showing that using training augmentations with AdaBN can be necessary for improving OOD accuracy.

What does this have to do with mixture distributions? For CIFAR-10 and TinyImageNet, standard random cropping is part of the training augmentation. See Figure 5 for an example of this augmentation. Therefore, at training time an image is visibly cropped with probability p and not cropped (or only slightly cropped) with probability $1-p$, leading to a mixture distribution. At test time p becomes 0. Furthermore, analogously to the conceptual example, test accuracy is high without feature alignment but becomes lower with alignment.



(a) Augmented (source) (b) Original (target)

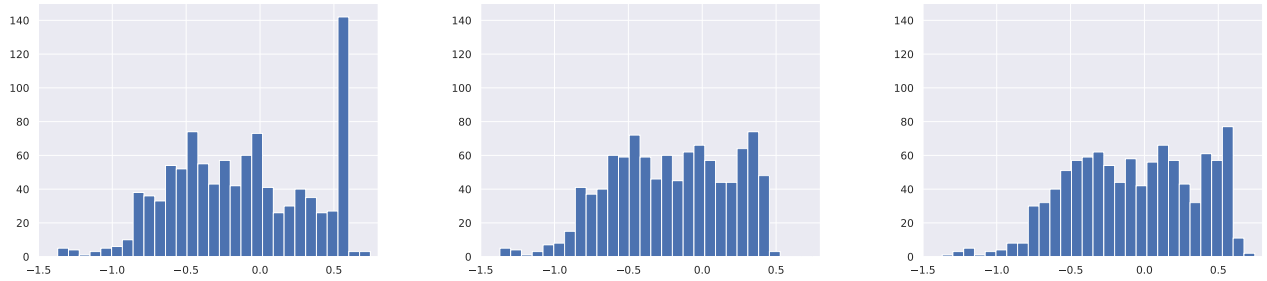
Figure 5: An example of the augmentation used to train our CIFAR-10 and TinyImageNet models. The source (training) distribution includes images like Figure 5a while the target (test) distribution only includes unaugmented examples like Figure 5b, causing a harmless distribution shift.

To better understand this distribution shift, we visualize the activations of a convolutional filter in the first Batch Normalization layer in Figure 6. These activations are for a network with the image in Figure 5 as its input. We find that cropping with padding causes a peak in the activations of this filter (Figure 6a) that disappears when we remove the augmentation (Figure 6b). Consequently, if one applies AdaBN to the test set, the mean activation for this filter on the test set is less than during training. In trying to correct for this, AdaBN increases all activations (Figure 6c). This type of shift in the distribution of activations, which we also find for other images and layers, appears responsible for the performance degradation when AdaBN is applied to the clean test set. In short, while reweighting the training examples doesn't hurt the model, AdaBN detects a change and renormalizes the activations when it shouldn't.

4. Understanding AdaBN & Discussion

We now explain when and why AdaBN can improve or degrade robustness. The best case scenario for AdaBN is when a shift transforms the activations in a convolutional channel, $\mathbf{x}_i \in \mathbb{R}^{d\times d}$, according to

$$\hat{\mathbf{x}}_i = a\mathbf{x}_i + b\cdot\mathbf{1}\mathbf{1}^T\,, \tag{2}$$



(a) Original Model, Train Aug (Source)

(b) Original Model, Test Aug (Target)

(c) Updated Model, Test Aug (Target)

Figure 6: Histograms of all activations for a single CIFAR-10 example and channel in the first Batch Normalization layer. This shows that features can be multimodal and that changes in the frequency of one mode (the spike of activations centered around 0.5 on the source distribution in Figure 6a, which is heavily downweighted on the target distibution in Figure 6b) can cause AdaBN to shift another mode (the bulk of the activations, which shift to the right in Figure 6c relative to Figure 6b) when it would be better not to change the other activations at all.

for each example i. Here, d is the spatial height and width, and $\mathbf{1}$ is the d-dimensional vector of all ones.

This is an affine function with coefficients shared across examples and spatial dimensions. Since this transformation corresponds to changing the mean and variance (shared across examples and spatial dimensions), AdaBN exactly inverts distribution shifts that affect the activations in this way. This characterization of AdaBN provides insight into both why it can degrade robustness for some shifts and why it improves robustness for others.

4.1. Why Can Aligning BN Statistics Hurt?

Equation (2) makes it clear why the failure modes we presented can occur. Because the coefficients a and b are fixed across examples and spatial dimensions, AdaBN relies on the implicit assumption that different samples and spatial locations are shifted in similar ways. While this is useful because it makes it possible for AdaBN to efficiently estimate the new mean and variance under the shift, it also means that AdaBN can degrade performance when this assumption is violated. Indeed, every failure mode in Section 3 violates this assumption. Concerningly, it may not be easy in practice to assess whether this assumption is being violated or not. This may make it difficult to trust methods like AdaBN in high-stakes applications involving arbitrary unforeseen distribution shifts for which we need high reliability.

4.2. Why Can Aligning BN Statistics Help?

In practice, methods like AdaBN that align first and second order activation statistics can yield state-of-the-art robustness [21, 27]. Moreover, by Equation (2), one can think of AdaBN as trying to invert a change in the scale and mean of activations. The empirical success of AdaBN, together with this interpretation of Equation (2), suggests that one of the main effects of some distribution shifts is to simply change the scale and mean of the network's activations in each hidden layer.

We now provide intuition for why this might be true. Consider a convolutional filter in any layer of a CNN. One can think of it as a feature detector that activates most for a certain input pattern. Suppose this pattern becomes more common for all inputs under the distribution shift. Then even if this pattern was strongly correlated with a specific class on the source distribution, it will only be weak evidence of that class on the target distribution. This suggests we should align the means; if every image becomes greener than it was before, and green is correlated with being a frog, then under the shift we should now consider "very green" to be evidence of a frog but "somewhat green" to be uninformative.

Normalizing the variance, on the other hand, intuitively corrects for simple changes in the scale of the activations of a convolutional filter. For instance, if a feature becomes obscured under shift, such as if edges become blurrier, a convolutional filter that was trained to detect that feature may output activations that are closer to zero, decreasing the variance of this activation. However, the next layer still expects its inputs to be in a certain range. In this situation, normalizing the variance may amplify the signal that does exist by increasing the scale of the activations.

In short, because changes in the prevalence of a pattern may result in simple changes in the activations of a feature detector for that pattern, normalization may be an effective way to partially undo the effects of some distribution shifts. This is the property that methods like AdaBN exploit.

4.3. When Does AdaBN Help the Most?

Geirhos et al. [9] argue that most ImageNet classifiers are overly reliant on the texture and style of images. This finding implies that most hidden layers in modern ImageNet classifiers capture low-level features such as texture or style more than they capture high-level features such as shape. Furthermore, AdaBN improves robustness by tweaking the activations of a trained network. This suggests that it is mostly "fixing" changes in style, since those are what activations mainly capture in the first place. Indeed, more abstract shifts in the distribution might not even register in the activations of the model because it was not trained to detect those sorts of features.

Relatedly, Li et al. [18] draw a connection between style transfer methods and domain adaptation. They show that simply aligning the Batch Normalization statistics between two images can be used as an effective method for style transfer. Similarly, AdaBN aligns the Batch Normalization statistics between two distributions. This suggests that we can also interpret AdaBN as doing style transfer between two distributions, mapping the style of the shifted target distribution back to that of the original source distribution.

These perspectives predict that AdaBN should improve accuracy the most on distribution shifts involving changes in style and local image statistics, at least for current models, but should not substantially change performance on distribution shifts that involve more high-level, abstract changes. These predictions are supported by the observation that AdaBN improves accuracy much more on ImageNet-C and Stylized ImageNet than ImageNetV2. Table 1 shows that AdaBN yields a relative accuracy improvement of 23% for ImageNet-C and 43% for Stylized ImageNet, shifts that almost exclusively involve changes in the style or texture of images, whereas it slightly degrades performance on ImageNetV2, a recollected version of ImageNet that should not have major differences in local image statistics. This observation is further supported by the results in Schneider et al. [27], which show that aligning BN statistics also does not help much with ImageNet-A [12] or ObjectNet [1], two other distribution shift benchmarks that, like ImageNetV2, do not primarily involve changes in local image statistics.

These findings provide evidence for the idea that AdaBN improves robustness because it performs a sort of neural style transfer between the source and target distributions. While this makes AdaBN particularly well suited for some types of shift, such as ImageNet-C and Stylized ImageNet, it also suggests that the lackluster performance of the method on other types of distribution shifts is an inherent limitation rather than one that can be easily fixed.

5. Conclusion

Unforeseen Distribution Shifts. Making systems robust under distribution shift is important for a wide range of applications [10]. UDA is considered a promising approach to this problem, but our results show that it must be used with care. For applications like self-driving cars, UDA methods should work even when applied to general, unforeseen distribution shifts. However, we find that aligning batch normalization statistics may actually degrade robustness on shifts that can arise in practice. These limitations call into question the practical utility of aligning batch normalization statistics to improve robustness, especially for use in high-stakes applications.

Learning Representations. We also find that AdaBN disproportionately improves robustness on distribution shifts that mainly involve changes in local image statistics, such as changes in style or texture. It cannot help as much on distribution shifts involving changes in higher-level features because it only tweaks the activations of a trained network, which may not capture information about the high-level features that changed. This limitation suggests that to improve robustness for more general distribution shifts, it may be necessary to focus on learning robust representations rather than on modifying the activations of trained networks.

On the other hand, UDA methods that require additional training typically do so at test time. This is too slow for applications such as autonomous vehicles for which it is necessary to make predictions efficiently. These drawbacks may make typical UDA methods a less promising approach to improving model robustness than other techniques that train models to have broadly robust feature representations, such as architectural changes [22] or data augmentation [11].

Future Work. To the best of our knowledge, there has been limited work on investigating how distribution shifts affect low-level network activations in a fine-grained way. Building on our work by analyzing these effects in more detail may yield additional insights into distribution shifts and the learned feature representations, and may help us develop better methods for improving robustness.

Furthermore, while there are theoretical justifications of feature alignment, they do not adequately explain when or why these methods work well in practice (Section 2). We conceptually and empirically addressed this in detail in the case of AdaBN, a particularly simple but effective method. Future work should more carefully identify when and why other methods for robustness are effective in practice.

Finally, we identified numerous drawbacks of current approaches to UDA. Future work should address these shortcomings to make these methods more useful and reliable for important applications.

References

[1] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *NeurIPS*, 2019. 8

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. In *Machine learning*, 2010. 2

[3] Shai Ben-David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *AISTATS*, 2010. 2

[4] Shai Ben-David and Ruth Urner. On the hardness of domain adaptation and the utility of unlabeled target samples. In *ALT*, 2012. 2

[5] Fabio Maria Cariucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *ICCV*, 2017. 1, 2

[6] Zhijie Deng, Yucen Luo, and Jun Zhu. Cluster alignment with a teacher for unsupervised domain adaptation. In *ICCV*, 2019.

[7] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 2

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016. 1, 2

[9] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019. 3, 8

[10] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. 1, 3, 8

[11] 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. In *ICLR*, 2020. 8

[12] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, 2021. 8

[13] Haoshuo Huang, Qixing Huang, and Philipp Krahenbuhl. Domain transfer through deep activation matching. In *ECCV*, 2018. 1

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1

[15] Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *ICML*, 2016. 1

[16] Fredrik D Johansson, Rajesh Ranganath, and David Sontag. Support and invertibility in domain-invariant representations. In *AISTATS*, 2019. 2, 3

[17] Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, Bill Freeman, and Gregory Wornell. Co-regularized alignment for unsupervised domain adaptation. In *NeurIPS*, 2018. 1

[18] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. In *IJCAI*, 2017. 8

[19] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 2018. 1

[20] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018. 1, 2

[21] Zachary Nado, Shreyas Padhy, D. Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020. 1, 7

[22] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018. 8

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 3

[24] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019. 1, 3

[25] Ievgen Redko, Nicolas Courty, Rémi Flamary, and Devis Tuia. Optimal transport for multi-source domain adaptation under target shift. In *AISTATS*, 2019. 2, 3

[26] Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulo, Nicu Sebe, and Elisa Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. In *CVPR*, 2019. 1

[27] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. 1, 7, 8

[28] Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *ICML*, 2017. 1

[29] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *ICLR*, 2018. 1

[30] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016. 1, 2

[31] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016. 1, 2

[32] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 2018. 1

[33] Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *NeurIPS*, 2019. 1, 2

[34] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 3

[35] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J Gordon. On learning invariant representation for domain adaptation. In *ICML*, 2019. 2, 3