SourceStein: Shrinking Test-Time Normalization Statistics toward the Source in Closed Form

Anonymous Authors[1]

Abstract

Test-time normalization makes a classifier with batch normalization more robust to distribution shift by re-estimating its normalization statistics on each test batch, without any training. However, current methods mix source and test batch statistics with a weight fixed in advance, which causes two drawbacks: (i) under a large shift, a small weight keeps stale source statistics; and (ii) under a small batch, a large weight copies noisy batch statistics. To address both, we propose **SourceStein**, an adaptive and stable rule that sets the mixing weight of each layer after seeing the batch, by James–Stein shrinkage of the batch statistics toward the source statistics. Specifically, SourceStein measures the shift of the batch in units of the noise expected from its size, and moves toward the batch only as far as the shift exceeds this noise. The resulting weight is closed-form, has no hyperparameters and, under a Gaussian model, never has a higher risk than the batch statistics. Experiments with two networks on common image corruptions show that SourceStein generally reduces the error of prior mixing rules for small batches, while matching them for large ones.

1. Introduction

Batch normalization (BN) (Ioffe & Szegedy, 2015) is part of most convolutional classifiers. Each BN layer normalizes its features with statistics collected on the training data, which ties the whole network to the training distribution. Under covariate shift (Schneider et al., 2020), such as fog or sensor noise at test time, these statistics no longer match the test features. As a result, the accuracy of a model trained on clean images decreases sharply (Hendrycks & Dietterich, 2019). Test-time normalization (You et al., 2021) removes much of this loss by re-estimating the BN statistics on the test data, with no labels and no training (Nado et al., 2020; Schneider et al., 2020). Because it changes only the statistics, it applies to any trained BN network and adds almost no computation. While effective, this fix depends on how many test inputs arrive together: streams and edge devices often deliver a handful of images at a time (Gong et al., 2022; Niu et al., 2023), and statistics computed from so few samples are unreliable (Schneider et al., 2020).

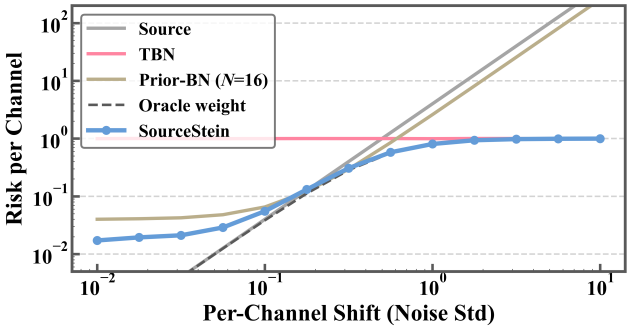


Figure 1. Fixed mixing weights fail at one end of the shift range. Risk of the estimated mean (per channel, in noise units, log scale) for batches of $n{=}4$ simulated samples with $C{=}64$ channels, as the shift increases. Source statistics and Prior-BN incur a shift bias that increases with the shift, while TBN incurs the full estimation variance at each shift; SourceStein follows the oracle weight across the range.

To handle small batches, current methods combine two estimates: the stored source statistics (Schneider et al., 2020) and the test batch statistics (Tomar et al., 2024). In our notation, a layer normalizes with the mean $\boldsymbol{\mu}^{\text{src}} + \lambda(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}})$, where $\hat{\boldsymbol{\mu}}$ is the mean of the n test samples and $\lambda \in [0, 1]$ is a mixing weight (the variance is treated alike). TBN (Nado et al., 2020) uses $\lambda{=}1$, Prior-BN (Schneider et al., 2020) uses $\lambda{=}n/(n{+}N)$ with a pseudo sample size N, and α-BN (You et al., 2021) uses a constant $\lambda{=}1{-}\alpha$. Each choice is set by a hyperparameter and the batch size, and works well in the setting it was tuned for. These rules have made test-time normalization a strong training-free baseline (Wang et al., 2021). Whether full, prior-weighted or constant, these rules share one design: the mixing weight is fixed before the test batch is seen. Consider a batch of

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

four foggy images (the running example of Figure 2): Prior-BN assigns it the same weight $4/(4{+}N)$ as a batch of four clean images, although only the foggy batch has moved away from the training data.

Intuitively, effective test-time statistics should be adaptive (*i.e.,* follow the shift of the test data) and stable (*i.e.,* not fluctuate with the few samples in a batch). However, a weight fixed in advance cannot be both. As shown in Figure 1 on simulated features, this design causes two limitations:

- **Stale Source Statistics.** When the shift is large and the weight is small, the mixed mean remains near the source mean. As a result, the normalization inherits a shift bias that increases with the square of the shift. This bias affects Prior-BN most at large N, whose weight is in fact optimal for only one shift size (see Section 3.1).
- **Noisy Batch Statistics.** When the batch is small and the weight is large, the mixed mean copies the sampling noise of $\hat{\boldsymbol{\mu}}$. Consequently, the normalization incurs an estimation variance that does not decrease with the shift. This variance makes TBN err on most clean inputs at batch size one (see Table 2).

To address these issues, we propose **SourceStein**, an adaptive and stable rule for test-time normalization. The key idea is to set the mixing weight after the test batch is seen, from the batch itself, by comparing how far the batch has moved with how far it would move by chance. In this way, the weight becomes a measurement of each batch rather than a setting chosen before deployment. However, this comparison requires the noise level of the batch statistics, which a single test image cannot provide. Moreover, a plug-in estimate of the best weight is itself noisy for small batches (Efron & Morris, 1973). To this end, we equip each BN layer with two parts. The first is a **shift score** $S = n\|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}}\|^2_{\boldsymbol{\Sigma}^{-1}}$, which whitens the observed shift with between-sample variances $\boldsymbol{\Sigma}$ stored from the training data. The second is a **Stein weight** $\hat{\lambda} = (1 - (C{-}2)/S)_+$ over the C channels of the layer, *i.e.,* the positive-part James–Stein estimator (James & Stein, 1961) with the source statistics as its target. The shift score keeps the statistics adaptive when the test data have moved, while the Stein weight keeps them stable when the move is within the noise. Under a Gaussian model, this rule never has a higher risk than TBN and remains close to the best fixed weight for the actual shift.

We then run experiments with two networks on CIFAR-10-C (Hendrycks & Dietterich, 2019) to test SourceStein. First, SourceStein generally reduces the error of TBN, Prior-BN and α-BN for batches of 1 to 16 images and matches them for larger batches (*cf.* Section 4.1). Second, its weight increases with the batch size and with the shift, as the analysis predicts (*cf.* Section 4.2). Third, adapting the variance and pooling the score over channels both contribute, and no pseudo sample size of Prior-BN matches SourceStein (*cf.* Section 4.3). Fourth, inside entropy minimization the rule helps one of the two networks (*cf.* Section 4.4). Overall, the rule replaces a hyperparameter that previously had to be tuned on labelled shifted data with a decision made from each batch. We highlight that SourceStein is adaptive and stable at the cost of two stored vectors per layer and no training.

2. Preliminary

Section 2.1 first defines test-time normalization. After that, Section 2.2 writes the existing rules as one family of mixed statistics and explains why small batches are hard for them.

2.1. Task Formulation

Fully test-time adaptation. We start from a classifier f trained on a source distribution $\mathcal{P}^{\text{src}}$ and deployed on a shifted target distribution $\mathcal{P}^{\text{tgt}}$ (Wang et al., 2021). Unlabelled test inputs arrive in batches $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^n$ of a small size n, and the model must predict each batch without labels, source data or further training. We consider the setting in which each batch is drawn independently from $\mathcal{P}^{\text{tgt}}$ and the model keeps no memory across batches, which isolates the effect of the batch size. This setting matches the evaluation of test-time normalization in prior work (Nado et al., 2020; Schneider et al., 2020), and it is the hardest one for batch statistics, since no evidence accumulates over time.

Batch normalization. Consider one BN layer with C channels (Ioffe & Szegedy, 2015). For a feature map $\mathbf{h} \in \mathbb{R}^{C\times H\times W}$, the layer outputs:

$$\text{BN}(\mathbf{h}) = \boldsymbol{\gamma} \odot \frac{\mathbf{h} - \boldsymbol{\mu}}{\sqrt{\mathbf{v} + \epsilon}} + \boldsymbol{\beta}, \tag{1}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^C$ are the learned affine parameters, $\boldsymbol{\mu}, \mathbf{v} \in \mathbb{R}^C$ are the mean and variance used for normalization (broadcast over the spatial positions), and ϵ is a small constant. After training, the layer stores the source statistics $\boldsymbol{\mu}^{\text{src}}$ and $\mathbf{v}^{\text{src}}$ (running averages over the training data), and the standard model uses them for each test input.

2.2. Mixed Statistics for Test-Time Normalization

Per-sample and batch statistics. For test sample i, let $\mathbf{m}_i \in \mathbb{R}^C$ be the spatial mean of its feature map and $\mathbf{u}_i \in \mathbb{R}^C$ the spatial mean of $(\mathbf{h}_i - \boldsymbol{\mu}^{\text{src}})^2$, its second moment about the source mean. The test batch statistics are then:

$$\begin{aligned} \hat{\boldsymbol{\mu}} &= \tfrac{1}{n}\textstyle\sum_{i=1}^n \mathbf{m}_i, \qquad \hat{\mathbf{u}} = \tfrac{1}{n}\textstyle\sum_{i=1}^n \mathbf{u}_i, \\ \hat{\mathbf{v}} &= \hat{\mathbf{u}} - (\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}})^2, \end{aligned} \tag{2}$$

where $\hat{\mathbf{v}}$ is the usual batch variance over all nHW positions, written through $\hat{\mathbf{u}}$ so that both statistics are averages of independent per-sample terms.

Mixing rules. Existing test-time normalization methods replace $\boldsymbol{\mu}$ in Equation 1 by a convex combination of the two statistics:

$$\boldsymbol{\mu}(\lambda) = \boldsymbol{\mu}^{\text{src}} + \lambda\,(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}}), \qquad \lambda \in [0,1], \tag{3}$$

and treat the variance in the same way. The source model is λ=0 and TBN (Nado et al., 2020) is λ=1. Prior-BN (Schneider et al., 2020) reads the source statistics as a prior worth N samples and sets $\lambda = n/(n+N)$, recommending $N \in [8, 128]$ for small batches. α-BN (You et al., 2021) mixes the means and standard deviations with a constant $\lambda = 1-\alpha$, with α=0.9 for classification. This weight does not depend on n, and α-BN keeps most of the source statistics even when a large batch would allow a precise estimate. Methods that additionally train the affine parameters, such as TENT (Wang et al., 2021), build on the same statistics. Other rules set λ through a schedule over the test stream (Mirza et al., 2022) or learn one weight per layer on augmented source data before deployment (Lim et al., 2023). Whether set by a hyperparameter, a schedule or a training phase, the weight is decided before the test batch arrives.

Noise of the batch statistics. The per-sample means $\mathbf{m}_i$ of a batch are independent draws. As a result, the batch mean $\hat{\boldsymbol{\mu}}$ has covariance $\text{diag}(\boldsymbol{\tau}^2)/n$, where $\boldsymbol{\tau}^2$ collects the between-sample variances of the channel means. For n=1, this noise equals the full spread of single images, and TBN normalizes each image by its own statistics, which can remove the global contrast and brightness that the classifier relies on. The same holds for $\hat{\mathbf{u}}$ with the between-sample variances $\boldsymbol{\rho}^2$ of the second moments. In contrast, the shift $\boldsymbol{\mu}^{\text{tgt}} - \boldsymbol{\mu}^{\text{src}}$, where $\boldsymbol{\mu}^{\text{tgt}} = \mathbb{E}_{\mathcal{P}^{\text{tgt}}}[\mathbf{m}_i]$ is the target mean, does not depend on n. Therefore, a good weight should compare the two quantities, which the rules above never do. Each of their weights depends only on n and a hyperparameter chosen in advance. Still, a rule that cannot tell a batch that has moved far from one that has moved by chance leaves the statistics either stale or noisy, which limits test-time normalization exactly when batches are small.

3. Methodology

In this section, we present **SourceStein**, an adaptive and stable rule that chooses the mixing weight of Equation 3 separately for each test batch and each BN layer. We first write the error of the mixed statistics as a sum of a shift bias and an estimation variance in Section 3.1, which yields the best mixing weight in closed form. After that, Section 3.2 estimates the unknown shift with a whitened shift score. Then, Section 3.3 turns the score into the Stein weight and states its risk guarantee. Finally, Section 3.4 applies the same rule to the variances and assembles the complete method.

3.1. Test-Time Statistics as a Shrinkage Problem

To choose λ for a given batch, we first state what a good mixed mean should achieve. Recall that $\boldsymbol{\mu}^{\text{tgt}}$ is the mean the layer would use with unlimited test data, and let $\boldsymbol{\delta} = \boldsymbol{\mu}^{\text{tgt}} - \boldsymbol{\mu}^{\text{src}}$ be the true shift and $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\tau}^2)$ the covariance of a single per-sample mean $\mathbf{m}_i$. The batch mean $\hat{\boldsymbol{\mu}}$ then has covariance $\boldsymbol{\Sigma}/n$. We measure the error of Equation 3 in the noise units of the layer:

$$\begin{aligned} R(\lambda) &= \mathbb{E}\, n \big\| \boldsymbol{\mu}(\lambda) - \boldsymbol{\mu}^{\text{tgt}} \big\|^2_{\boldsymbol{\Sigma}^{-1}} \\ &= \underbrace{(1-\lambda)^2\,\Delta}_{\text{shift bias}} + \underbrace{\lambda^2\, C}_{\text{estimation variance}}, \end{aligned} \tag{4}$$

where $\|\mathbf{x}\|^2_{\boldsymbol{\Sigma}^{-1}} = \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x}$ and $\Delta = n\|\boldsymbol{\delta}\|^2_{\boldsymbol{\Sigma}^{-1}}$ is the whitened shift. Specifically, the first term is the shift bias of keeping the source statistics, and the second is the estimation variance of trusting n samples. Setting the derivative to zero yields the best weight of the family:

$$\lambda^\star = \frac{\Delta}{\Delta + C}, \qquad R(\lambda^\star) = \frac{C\,\Delta}{\Delta + C}. \tag{5}$$

This oracle weight moves toward the batch as the shift or the batch increases, and remains near the source when the shift is small compared with the noise. In contrast, the weight $n/(n+N)$ of Prior-BN equals $\lambda^\star$ only when $\|\boldsymbol{\delta}\|^2_{\boldsymbol{\Sigma}^{-1}} = C/N$, *i.e.,* for one shift size per pseudo sample size. TBN incurs $R(1) = C$ at each shift, and the source model incurs $R(0) = \Delta$, which is unbounded in the shift.

3.2. Adaptive Mixing via a Whitened Shift Score

To counter stale source statistics, the weight must respond to the shift Δ, which is unknown at test time. We estimate it from the test batch statistics through the observed shift $\hat{\boldsymbol{\delta}} = \hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}}$ through the shift score:

$$S = n\,\big\|\hat{\boldsymbol{\delta}}\big\|^2_{\boldsymbol{\Sigma}^{-1}} = n \sum_{c=1}^{C} \frac{(\hat{\mu}_c - \mu_c^{\text{src}})^2}{\tau_c^2}, \tag{6}$$

whose expectation is $\mathbb{E}[S] = \Delta + C$: the true shift plus the noise of n samples. Specifically, the score requires the noise level $\boldsymbol{\tau}^2$, which a batch of one image cannot estimate. Therefore, we measure it once on the training data, as the variance of the per-sample means $\mathbf{m}_i$ over 5,000 training images, and store it together with $\boldsymbol{\mu}^{\text{src}}$. Under this choice, a shift that is large compared with the training spread of single images drives S far above C, and the weight can follow it. In contrast, a clean batch has $\mathbb{E}[S] = C$ up to the train–test gap of the source statistics, and its score remains close to the number of channels.

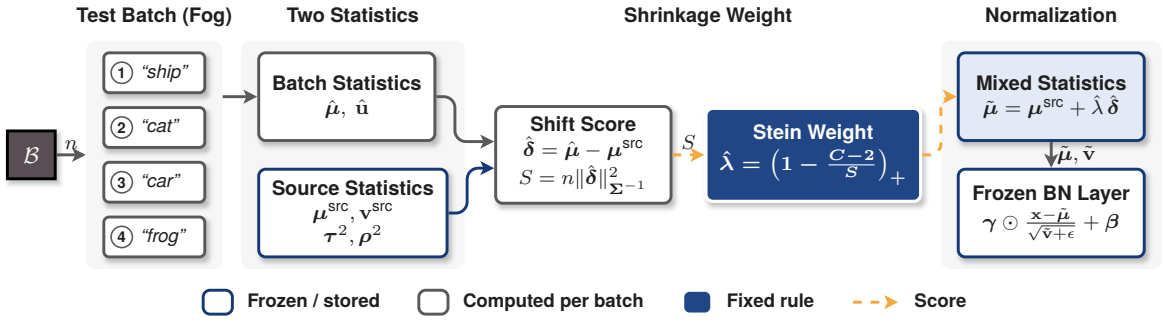


Figure 2. Overview of SourceStein at one BN layer. A test batch of n fog-corrupted images yields the batch statistics $\hat{\mu}, \hat{\mathbf{u}}$, while the frozen layer stores the source statistics $\mu^{\text{src}}, \mathbf{v}^{\text{src}}$ and the between-sample variances τ^2, ρ^2 measured once on training data. (1) The shift score S measures the observed shift $\hat{\delta}$ in units of the noise expected from n samples. (2) The Stein weight $\hat{\lambda}$ keeps only the part of the shift that exceeds this noise. (3) The mixed statistics replace the stored ones inside the unchanged BN layer.

Choice of the noise level. The source noise level is an assumption: a corruption can change the spread of single images as well as their mean. We keep it because it is available for each batch size, including $n{=}1$, while a noise level estimated from n test samples is itself a noisy ratio for small n. Section 4.3 compares both choices, and Appendix E shows what a misestimated noise level costs in simulation.

3.3. Stable Mixing with the Stein Weight

With the shift measured by S, the plug-in of Equation 5 would be $1 - C/S$. However, S is noisy when n is small, and a plug-in weight inherits this noise exactly where the estimation variance matters most. Therefore, we rely on the following classical result, in which the shrinkage target is the source mean:

Lemma 1 (James–Stein Estimator (James & Stein, 1961))**.** *Let $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\theta}, \mathbf{I}_C)$ with $C \geq 3$, let $\mathbf{a} \in \mathbb{R}^C$ be fixed, and let $s = \|\mathbf{z} - \mathbf{a}\|^2$. Then $\hat{\boldsymbol{\theta}} = \mathbf{a} + (1 - (C-2)/s)(\mathbf{z} - \mathbf{a})$ has risk $\mathbb{E}\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|^2 = C - (C-2)^2\,\mathbb{E}[1/s]$, and clipping the factor at zero does not increase the risk (Efron & Morris, 1973).*

Applying the lemma to the whitened batch mean $\mathbf{z} = \sqrt{n}\,\boldsymbol{\Sigma}^{-1/2}\hat{\boldsymbol{\mu}}$ with $\mathbf{a} = \sqrt{n}\,\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}^{\text{src}}$, for which $s = S$, yields the Stein weight:

$$\hat{\lambda} = \left(1 - \frac{C-2}{S}\right)_+, \tag{7}$$

where $(x)_+ = \max(x, 0)$. As a result, the weight is zero until the shift score exceeds $C - 2$ and approaches one as the shift dominates the noise.

One weight per layer. The C channels of a layer share one weight, because the lemma pools their evidence: a shift that is small in each channel but present in many of them still increases S. In contrast, a separate weight per channel would base each decision on a single noisy value, and the guarantee below requires $C \geq 3$. However, the price is that channels with different shifts receive the same weight, which Section 4.3 examines. The risk of the Stein weight compares with the two fixed extremes and with the oracle as follows:

Proposition 1. *Assume $\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\boldsymbol{\mu}^{tgt}, \boldsymbol{\Sigma}/n)$ with $C \geq 3$. Then the Stein weight satisfies $R(\hat{\lambda}) \leq C - (C-2)^2/(C+\Delta)$, which is **below the risk** C **of TBN** for each shift, and $R(\hat{\lambda}) - R(\lambda^\star) \leq 4(C-1)/(C+\Delta) < 4$.*

Specifically, the first bound follows from Lemma 1 and Jensen's inequality $\mathbb{E}[1/S] \geq 1/\mathbb{E}[S]$, and the second subtracts Equation 5 (see Appendix B for the proof). Measured per channel, the excess over the oracle is below $4/C$, which is small for layers with 16 to 128 channels.

Relation to Prior-BN. Each Stein weight corresponds to a pseudo sample size $\hat{N} = n(1 - \hat{\lambda})/\hat{\lambda}$ of Prior-BN, chosen per layer and per batch. Specifically, a batch whose shift score is close to C receives a very large $\hat{N}$ and keeps the source statistics, while a batch with a large score receives a small $\hat{N}$ and moves toward its own statistics. Prior-BN, in contrast, uses one N for all layers, batches and corruptions.

Discussion. The Stein weight has an empirical Bayes reading (Efron & Morris, 1973): $(C-2)/S$ estimates the share of the observed shift that is noise, and $\hat{\lambda}$ keeps the rest. Moreover, the choice of the shrinkage target matters here. Prior uses of Stein shrinkage in normalization layers shrink toward a centre computed from the same batch during training (Khoshsirat & Kambhamettu, 2023; Ivolgina et al., 2025), which reduces noise but cannot correct a shift. SourceStein instead shrinks toward the source statistics, so that the target contains the knowledge of the training data and the batch only has to supply the evidence of a shift.

3.4. SourceStein

Combining the shift score with the Stein weight, SourceStein treats the variance in the same way as the mean. The

Algorithm 1: Normalization with SourceStein

Input: test features $\{\mathbf{h}_i\}_{i=1}^n$ at one BN layer; stored $\boldsymbol{\mu}^{\text{src}}, \mathbf{v}^{\text{src}}, \boldsymbol{\tau}^2, \boldsymbol{\rho}^2, \boldsymbol{\gamma}, \boldsymbol{\beta}$.

// batch statistics (Equation 2)

$\mathbf{m}_i \leftarrow \text{mean}_{hw}\, \mathbf{h}_i,\ \mathbf{u}_i \leftarrow \text{mean}_{hw}(\mathbf{h}_i - \boldsymbol{\mu}^{\text{src}})^2$

$\hat{\boldsymbol{\mu}} \leftarrow \text{mean}_i\, \mathbf{m}_i,\ \hat{\mathbf{u}} \leftarrow \text{mean}_i\, \mathbf{u}_i$

// shift scores and Stein weights

$S_\mu \leftarrow n \sum_c (\hat{\mu}_c - \mu_c^{\text{src}})^2 / \tau_c^2$

$S_u \leftarrow n \sum_c (\hat{u}_c - v_c^{\text{src}})^2 / \rho_c^2$

$\hat{\lambda}_\mu \leftarrow (1 - \frac{C-2}{S_\mu})_+,\ \hat{\lambda}_u \leftarrow (1 - \frac{C-2}{S_u})_+$ **Stein**

// mixed statistics (Equation 8)

$\tilde{\boldsymbol{\mu}} \leftarrow \boldsymbol{\mu}^{\text{src}} + \hat{\lambda}_\mu(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}})$

$\tilde{\mathbf{u}} \leftarrow \mathbf{v}^{\text{src}} + \hat{\lambda}_u(\hat{\mathbf{u}} - \mathbf{v}^{\text{src}})$

$\tilde{\mathbf{v}} \leftarrow \max(\tilde{\mathbf{u}} - (\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}})^2, 10^{-5})$

return $\boldsymbol{\gamma} \odot (\mathbf{h}_i - \tilde{\boldsymbol{\mu}})/\sqrt{\tilde{\mathbf{v}} + \epsilon} + \boldsymbol{\beta}$ for each i.

per-sample second moments $\mathbf{u}_i$ have mean $\mathbf{v}^{\text{src}}$ on source data and between-sample variances $\boldsymbol{\rho}^2$, which we store with $\boldsymbol{\tau}^2$. For each BN layer and each test batch, SourceStein computes:

$$\begin{aligned} \tilde{\boldsymbol{\mu}} &= \boldsymbol{\mu}^{\text{src}} + \hat{\lambda}_\mu\,(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}}), \\ \tilde{\mathbf{u}} &= \mathbf{v}^{\text{src}} + \hat{\lambda}_u\,(\hat{\mathbf{u}} - \mathbf{v}^{\text{src}}), \\ \tilde{\mathbf{v}} &= \max\big(\tilde{\mathbf{u}} - (\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}^{\text{src}})^2,\ 10^{-5}\big), \end{aligned} \tag{8}$$

where $\hat{\lambda}_\mu$ is Equation 7 with $(\hat{\boldsymbol{\mu}}, \boldsymbol{\mu}^{\text{src}}, \boldsymbol{\tau}^2)$ and $\hat{\lambda}_u$ is the same rule with $(\hat{\mathbf{u}}, \mathbf{v}^{\text{src}}, \boldsymbol{\rho}^2)$. The layer then normalizes with $\tilde{\boldsymbol{\mu}}$ and $\tilde{\mathbf{v}}$ in Equation 1, and the next layer sees the adapted features. When both weights are one, Equation 8 is equivalent to the batch statistics of Equation 2; when both are zero, it is equivalent to the source model. Algorithm 1 lists all steps of SourceStein for one layer.

Cost. SourceStein requires one pass over 5,000 unlabelled training images to store $\boldsymbol{\tau}^2$ and $\boldsymbol{\rho}^2$, and no labels, gradients or hyperparameters. At test time it adds $O(nCHW)$ operations per layer, the same order as computing the batch statistics; further implementation details are in Appendix C.

4. Experiments

In this section, we examine SourceStein through the research questions below:

- **RQ1:** (Performance) Does SourceStein reduce the corruption error of existing mixing rules across batch sizes, and does it keep the error on clean data?
- **RQ2:** (Mechanism) How does the Stein weight balance the shift bias and the estimation variance as the batch size and the shift change?

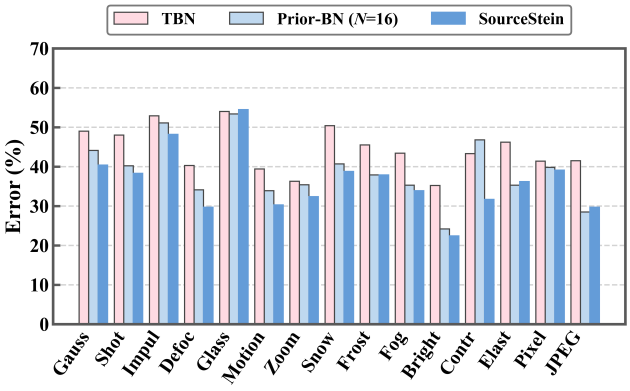


Figure 3. Error per corruption on CIFAR-10-C at severity 5 for ResNet-14 and batches of n=4 images. SourceStein is lowest on 12 of the 15 corruptions, with the largest gain on contrast.

- **RQ3:** (Ablation Study) What do pooling, variance shrinkage and the source noise level contribute, and can any fixed pseudo sample size match SourceStein?
- **RQ4:** (Extension) Can SourceStein replace the batch statistics inside entropy minimization?

Models. We train two BN networks from scratch on the CIFAR-10 training set on CPU: CNN-8, a VGG-style network (Simonyan & Zisserman, 2015) with six convolution and BN layers (86.54% clean test accuracy), and ResNet-14, a residual network (He et al., 2016) with fifteen BN layers (82.56%). Both use standard augmentation and SGD; Appendix C reports the training recipe.

Evaluation. We test on CIFAR-10-C (Hendrycks & Dietterich, 2019), which applies 15 common corruptions to the CIFAR-10 test set, at the highest severity 5 (severity 3 is reported in Appendix D). For each corruption, we draw a fixed random subset of 1,000 test images, shuffle it into batches of size $n \in \{1, 2, 4, 8, 16, 64, 200\}$, and report the top-1 error averaged over the 15 corruptions. All methods see the same images in the same order, and the statistics are recomputed for each batch without memory across batches.

Baselines. We compare with four rules of Equation 3: the Source model, TBN (Nado et al., 2020), Prior-BN (Schneider et al., 2020) with the pseudo sample size N=16, and α-BN (You et al., 2021) with α=0.9, as recommended by their authors. As a privileged reference, Oracle-N picks the best $N \in \{2, 4, \ldots, 256\}$ of Prior-BN for each network and batch size on the test results themselves. All methods share the trained weights, and differences come from the normalization statistics alone.

4.1. (RQ1) Error Across Batch Sizes

Table 1 reports the corruption error of each rule, Figure 4 shows it as a function of the batch size, and Table 2 reports the error on clean test images; Appendix D reports the per-

Table 1. Error (%) on CIFAR-10-C at severity 5, averaged over 15 corruptions, for test batches of n images. The best result in each column of a block is **bold**; the grey Oracle-N row tunes Prior-BN on the test results and is not ranked.

Network	Method	Test Batch Size n 1	2	4	8	16	64	200
ResNet-14	Source	52.06	52.06	52.06	52.06	52.06	52.06	52.06
	TBN (Nado et al., 2020)	89.19	55.06	44.45	37.80	33.99	31.06	30.40
	Prior-BN (Schneider et al., 2020)	48.57	43.67	38.71	35.26	32.97	31.01	**30.37**
	α-BN (You et al., 2021)	45.89	43.95	43.06	42.33	41.96	41.74	41.67
	Oracle-N (privileged)	43.53	40.65	37.52	35.26	32.97	30.85	30.21
	SourceStein (Ours)	**45.34**	**39.71**	**36.23**	**33.99**	**32.50**	**31.00**	30.41
CNN-8	Source	50.01	50.01	50.01	50.01	50.01	50.01	50.01
	TBN (Nado et al., 2020)	89.81	53.09	41.33	34.47	31.03	**28.01**	**27.25**
	Prior-BN (Schneider et al., 2020)	47.21	43.84	39.18	34.44	31.11	28.51	27.55
	α-BN (You et al., 2021)	45.61	43.99	43.21	42.63	42.63	42.28	42.11
	Oracle-N (privileged)	43.53	39.42	35.81	32.87	30.80	28.04	27.29
	SourceStein (Ours)	**43.93**	**37.89**	**34.65**	**32.23**	**30.43**	28.32	27.41

Table 2. Error (%) on 1,000 clean CIFAR-10 test images. The best result in each column is **bold**.

Method	ResNet-14 n=1	n=4	n=16	CNN-8 n=1	n=4	n=16
Source	**17.1**	**17.1**	**17.1**	13.9	13.9	13.9
TBN	88.5	32.4	22.8	89.9	29.7	17.8
Prior-BN	17.7	19.7	20.3	**13.6**	15.0	15.3
α-BN	18.2	17.8	18.1	14.3	**13.8**	**13.9**
SourceStein	19.3	18.4	18.3	15.0	14.5	14.0

Table 3. Inference time (ms per image) on one CPU thread for fog images at severity 5, averaged over three passes over 400 images.

Method	ResNet-14 n=1	n=16	n=200	CNN-8 n=1	n=16	n=200
Source	0.76	0.65	0.85	0.55	0.86	0.88
TBN	1.06	0.73	0.90	0.68	0.85	0.95
Prior-BN	1.10	0.73	0.89	0.69	0.94	0.92
SourceStein	1.48	0.78	0.88	0.85	0.83	0.95

corruption results. We have the following observations:

- **SourceStein has the lowest error for batches of 1 to 16 images.** As shown in Table 1, SourceStein reduces the error of the best unprivileged baseline at each $n \leq 16$ on both networks. The largest reductions are 3.96 points on ResNet-14 and 5.95 points on CNN-8, both at n=2. Moreover, it is below the privileged Oracle-N for n from 2 to 16, although Oracle-N tunes one pseudo sample size per batch size on the test results. We attribute this gain to the mixing weight of SourceStein, which differs per layer and per batch, while any N fixes one weight for all of them. At n=1, Oracle-N remains lower, by 1.81 and 0.40 points. Additionally, the gain is spread over most corruptions: as shown in Figure 3, SourceStein is lowest on 12 of 15 corruptions at n=4 and is above Prior-BN mainly on glass blur, elastic transform and JPEG.
- **For large batches, the rules converge.** At $n \geq 64$, SourceStein, TBN and Prior-BN are within 0.5 points of each other, with TBN slightly lower on CNN-8. This is expected, since the estimation variance vanishes as n increases and each reasonable weight approaches one. In contrast, α-BN keeps 90% of the source statistics at each n and remains above 41% error.

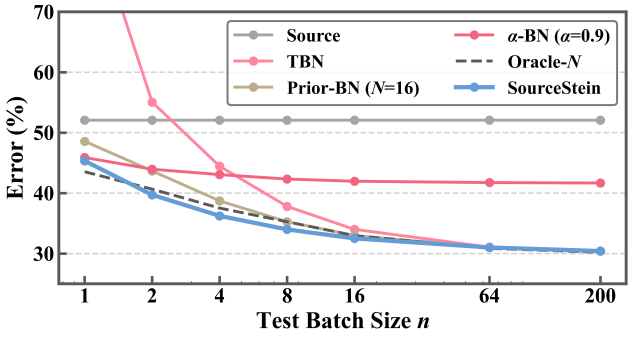
SourceStein remains close to the source model on clean data. Without a shift, any move toward the batch only adds estimation variance. As shown in Table 2, SourceStein remains within 2.2 points of the Source error at each batch size, while TBN loses up to 76 points and Prior-BN up to 3.7 points. Such results show that the Stein weight is stable when there is nothing to adapt to, at a small cost compared with α-BN, whose fixed weight favours clean data.

SourceStein adds little inference time. As shown in Table 3, SourceStein costs at most 0.42 ms more per image than TBN, at n=1 on ResNet-14, and is within the timing noise of TBN for larger batches. This additional work consists of two weighted sums over the channels of each layer, which is small compared with the convolutions.

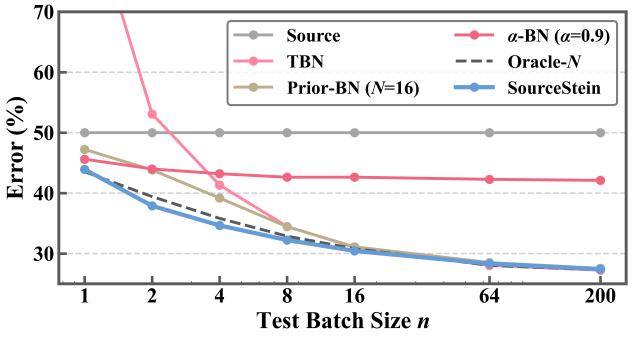
SourceStein also helps under milder shifts. To check that the gains are not tied to the strongest corruptions, we repeat the evaluation at severity 3. As shown in Table 4, SourceStein has the lowest error for n=4 and n=16 on both networks, while α-BN is lower for single images. Under a mild shift, a single image contains little evidence and the shift bias of the source statistics is small, which favours a weight close to zero; Appendix D.1 reports all batch sizes.

4.2. (RQ2) Study of the Stein Weight

Given that SourceStein reduces the error, we next test whether its weight behaves as Section 3.1 predicts, *i.e.,* whether it balances the shift bias and the estimation variance.



(a) ResNet-14.



(b) CNN-8.

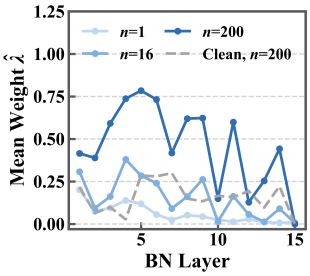
Figure 4. Error on CIFAR-10-C at severity 5 as the test batch size increases (log scale). SourceStein is the lowest unprivileged rule up to n=16 on both networks and joins TBN and Prior-BN for large batches.

Table 4. Error (%) on CIFAR-10-C at severity 3, averaged over 15 corruptions. The best result in each column is **bold**.

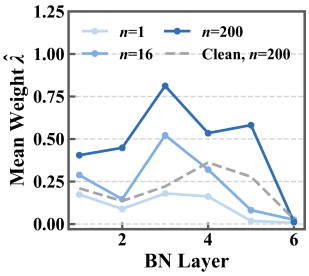
Method	ResNet-14			CNN-8		
	n=1	n=4	n=16	n=1	n=4	n=16
Source	35.91	35.91	35.91	32.73	32.73	32.73
TBN	89.21	38.56	28.12	89.97	34.67	24.47
Prior-BN	33.35	29.09	26.87	30.69	26.51	23.25
α-BN	**31.88**	30.35	29.50	**29.53**	28.43	27.82
SourceStein	34.24	**28.21**	**25.97**	31.30	**25.70**	**22.95**

Shift bias. The analysis predicts that the weight increases with the whitened shift Δ, which increases with n for a fixed corruption. As shown in Figure 5, the mean weight on corrupted data increases from 0.06 at n=1 to 0.46 at n=200 on ResNet-14, and from 0.11 to 0.47 on CNN-8. These weights are 1.5 to 3.5 times larger than on clean data at the same batch size. This is because a larger batch makes the same shift more visible above the noise. Moreover, the weights differ strongly across layers, which a single pseudo sample size cannot express.

Estimation variance. To isolate the estimator from the network, we simulate per-sample means with C=64 channels and a known noise level, as in Figure 1. For batch sizes from 1 to 64, the risk of SourceStein is at most 0.023 per channel above that of the oracle weight and never above that of TBN, consistent with Proposition 1. Figure 6 breaks

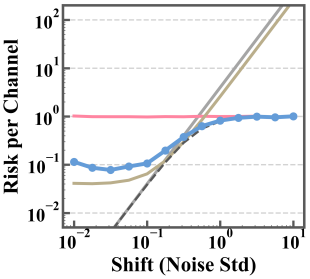


(a) ResNet-14.

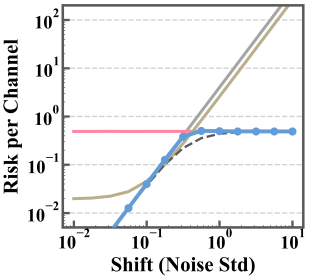


(b) CNN-8.

Figure 5. Stein weight $\hat{\lambda}_\mu$ per BN layer, averaged over test batches and the 15 corruptions at severity 5, for three batch sizes; the dashed line is clean data at n=200.



(a) Student-t_3 noise.



(b) Noise 0.7× the stored level.

Figure 6. Simulated risk per channel at n=4 when the assumptions of Proposition 1 fail, with the legend of Figure 1.

the two assumptions of the proposition. With heavy-tailed noise (6a), SourceStein remains below TBN at each shift and at most 0.114 per channel above the oracle. When the true noise is smaller than the stored level (6b), the score understates the evidence for a shift, and SourceStein exceeds TBN at 6 of 14 shift sizes; Appendix E adds further settings.

These results indicate that the gains of Table 1 come from a weight that follows the shift while remaining near zero when the batch contains little evidence.

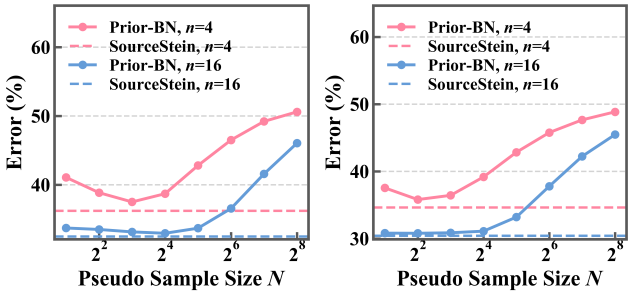
4.3. (RQ3) Ablation Study

To examine which parts of SourceStein matter, we remove one design choice at a time and report the results in Table 5.

Impact of each component. The first variant replaces the pooled Stein weight by a separate weight per channel (w/o pooling). The next two reuse the mean weight for the variance (w/o variance weight) or keep the source variance (w/o variance adaptation). The last estimates $\boldsymbol{\tau}^2$ and $\boldsymbol{\rho}^2$ from the test batch (w/ batch noise level), with the source values at n=1. We have the following observations: **adapting the variance matters most, and pooling helps mainly for small batches.** Keeping the source variance increases the error by 4 to 18 points at each batch size. This suggests that much of the corruption shift lies in the feature scale. Moreover, per-channel weights are worse at $n \leq 4$

Table 5. Ablation of SourceStein: error (%) on CIFAR-10-C at severity 5. The best result in each column of a block is **bold**.

	Variant	Test Batch Size n 1	4	16	64
ResNet-14	SourceStein	**45.34**	**36.23**	**32.50**	31.00
	w/o pooling	46.16	36.51	32.69	31.01
	w/o variance weight	46.41	37.28	32.76	31.21
	w/o variance adaptation	49.69	44.78	40.76	37.85
	w/ batch noise level	**45.34**	38.97	32.77	**30.95**
CNN-8	SourceStein	43.93	**34.65**	30.43	28.32
	w/o pooling	46.05	35.02	**30.16**	28.11
	w/o variance weight	**43.50**	36.13	31.68	29.49
	w/o variance adaptation	51.39	47.51	46.25	46.33
	w/ batch noise level	43.93	37.99	30.26	**28.09**



(a) ResNet-14. *(b)* CNN-8.

Figure 7. Error of Prior-BN over its pseudo sample size N for n=4 and n=16; dashed lines mark SourceStein at the same batch size.

on both networks but slightly better at $n \geq 16$ on CNN-8, consistent with pooling being most useful when each channel contains little evidence. Additionally, estimating the noise level from the test batch statistics hurts at n=4 by 2.7 to 3.3 points, since a variance from four samples is itself noisy.

Comparison with each pseudo sample size. Figure 7 varies N of Prior-BN from 2 to 256. At n=4 and n=16, no value of N reaches the error of SourceStein on either network, and the best N itself moves with the batch size and the network (from 2 to 64; Appendix C). Such results show that the per-batch weight of SourceStein replaces a hyperparameter that would otherwise require labelled shifted data.

4.4. (RQ4) Combination with Entropy Minimization

Finally, we test whether SourceStein can serve as the normalization inside TENT (Wang et al., 2021), which updates the BN affine parameters online by entropy minimization. We take one Adam step per batch with learning rate 10^{-3} and reset the model for each corruption.

As shown in Table 6, we have the following observation: **the gain of SourceStein extends to entropy minimization on ResNet-14 but not on CNN-8.** TENT with SourceStein is the best of the three rules on ResNet-14 for $n \leq 16$,

Table 6. Error (%) of TENT with three normalization rules on CIFAR-10-C at severity 5. The best result in each column is **bold**.

Normalization	ResNet-14 n=4	n=16	n=64	CNN-8 n=4	n=16	n=64
TBN	45.15	33.72	31.07	**42.85**	**30.99**	**27.93**
Prior-BN	64.17	33.12	**30.86**	66.31	31.85	28.42
SourceStein	**43.82**	**33.09**	31.10	44.98	31.31	28.49

while TENT with TBN is lower on CNN-8 at each batch size. A likely reason is that TENT already adapts the affine parameters to the test data. The gradient steps may then not correct the part of the shift bias that SourceStein leaves in the statistics, while TBN removes the full shift before them. Prior-BN with N=16 collapses at n=4 on both networks, which indicates that a fixed weight can interact badly with online updates. Moreover, all three rules use the learning rate of Wang et al. (2021), and a learning rate tuned for each rule may change this ranking. Overall, SourceStein is most useful as a training-free rule, and its combination with gradient-based adaptation remains network-dependent.

5. Limitations

This work still has a few limitations. On the one hand, the guarantee of Proposition 1 assumes Gaussian batch means with a noise level measured on source data, and Oracle-N remains lower at batch size one. On the other hand, we evaluate two small networks on synthetic corruptions, and whether the rule helps on larger models, natural shifts or non-i.i.d. streams is still unknown.

6. Conclusion

Test-time normalization methods mix source statistics and test batch statistics with a mixing weight fixed in advance, which leaves stale source statistics under large shifts and noisy batch statistics under small batches. To address both, this work presented SourceStein, an adaptive and stable rule that sets the weight of each layer from the batch by James–Stein shrinkage toward the source statistics. We showed that, under a Gaussian model, its risk never exceeded that of the batch statistics and remained close to the best fixed weight, balancing the shift bias and the estimation variance. Experiments on common corruptions showed that SourceStein generally reduced the error of prior mixing rules for small batches while matching them for large ones. We will further explore shrinkage targets that combine the source statistics with earlier test batches.

AI Disclosure Statement

This submission was produced with substantive use of a generative AI agent (a large language model with code-execution tools). The agent proposed the research idea, wrote the experimental code, ran all experiments, analysed the results, drew the figures and tables, searched and checked the literature, and drafted the manuscript. Each number in the paper is read from the raw output files of the released code, and each reference was compared with its arXiv or publisher record. The authors are responsible for reviewing all AI-generated content, including code, results, mathematical claims and citations, and take full responsibility for the paper.

Impact Statement

This work aims to make test-time normalization of deployed image classifiers more adaptive and stable when only a few test inputs arrive at a time, which can help applications such as on-device perception under changing sensors or weather. We expect its main societal effect to be positive, since a model that degrades less under distribution shift makes fewer silent errors. One risk is that a lower error under synthetic corruptions may be read as robustness to all real-world shifts, which our experiments do not establish.

References

Efron, B. and Morris, C. Stein's estimation rule and its competitors – an empirical Bayes approach. *Journal of the American Statistical Association*, 68(341):117–130, 1973.

Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., and Lee, S.-J. NOTE: Robust continual test-time adaptation against temporal correlation. In *NeurIPS*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hendrycks, D. and Dietterich, T. G. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Ivolgina, S., Fletcher, P. T., and Vemuri, B. C. Admissibility of Stein shrinkage for batch normalization in the presence of adversarial attacks. *CoRR*, abs/2507.08261, 2025.

James, W. and Stein, C. Estimation with quadratic loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 361–379, 1961.

Khoshsirat, S. and Kambhamettu, C. Improving normalization with the James-Stein estimator. *CoRR*, abs/2312.00313, 2023.

Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. Revisiting batch normalization for practical domain adaptation. *CoRR*, abs/1603.04779, 2016.

Lim, H., Kim, B., Choo, J., and Choi, S. TTN: A domain-shift aware batch normalization in test-time adaptation. In *ICLR*, 2023.

Mirza, M. J., Micorek, J., Possegger, H., and Bischof, H. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *CVPR*, 2022.

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., and Snoek, J. Evaluating prediction-time batch normalization for robustness under covariate shift. *CoRR*, abs/2006.10963, 2020.

Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., and Tan, M. Towards stable test-time adaptation in dynamic wild world. In *ICLR*, 2023.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Tomar, D., Vray, G., Thiran, J.-P., and Bozorgtabar, B. Unmixing test-time normalization statistics: Combatting label temporal correlation. In *ICLR*, 2024.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B. A., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

You, F., Li, J., and Zhao, Z. Test-time batch statistics calibration for covariate shift. *CoRR*, abs/2110.04065, 2021.

Zhao, B., Chen, C., and Xia, S.-T. DELTA: Degradation-free fully test-time adaptation. In *ICLR*, 2023.

A. Related Works

A.1. Test-Time Normalization

Re-estimating BN statistics on target data is a simple and widely adopted response to covariate shift. AdaBN (Li et al., 2016) replaces the source statistics by target-domain statistics for domain adaptation, and prediction-time BN (Nado et al., 2020) and the covariate shift adaptation of Schneider et al. (2020) recompute them on each test batch, Schneider et al. (2020) with a pseudo sample size N that mixes in the source statistics. α-BN (You et al., 2021) mixes the two with a constant weight, DUA (Mirza et al., 2022) updates them with a decaying momentum over the stream, TTN (Lim et al., 2023) learns a per-layer interpolation in a post-training phase on source data, and UnMix-TNS (Tomar et al., 2024) and NOTE (Gong et al., 2022) handle test streams whose labels are correlated in time. While effective, such rules fix how much they trust a test batch in advance (by a hyperparameter, a schedule or weights learned on source data), and they remain exposed to stale source statistics under large shifts and to noisy batch statistics under small batches.

A.2. Test-Time Adaptation with Gradient Updates

A second line updates model parameters on the test data. TENT (Wang et al., 2021) minimizes the prediction entropy with respect to the BN affine parameters, SAR (Niu et al., 2023) reports that BN makes such updates unstable for small batches and filters samples with large gradients, and DELTA (Zhao et al., 2023) renormalizes test-time BN statistics and reweights classes to remove biased updates. These methods all rely on some estimate of the normalization statistics, and Section 4.4 shows that the choice of this estimate changes their error. However, how to set the statistics of each batch without any hyperparameter remains largely underexplored in this line.

A.3. Shrinkage Estimation in Normalization Layers

The James–Stein estimator (James & Stein, 1961) improves the sample mean of three or more Gaussian means by shrinking it toward a fixed point, and the empirical Bayes view of Efron & Morris (1973) explains the shrinkage factor as an estimate of the signal-to-noise ratio. Recent work applies Stein shrinkage inside normalization layers during training: Khoshsirat & Kambhamettu (2023) shrink the batch means and variances of each layer toward their cross-channel average, and Ivolgina et al. (2025) prove that such shrinkage dominates the sample estimates under sub-Gaussian adversarial perturbations. These methods shrink toward a centre computed from the same batch, during training, to reduce noise rather than to correct a shift. To address these gaps, we propose SourceStein, which shrinks the test batch statistics toward the source statistics at test time, so that one closed-form weight per layer is adaptive and stable: it follows the shift when the batch contains evidence for it and keeps the source statistics otherwise.

B. Proof of Proposition 1

Setup. Our goal is to bound the risk of Equation 4 at the Stein weight. Let $\mathbf{z} = \sqrt{n}\,\boldsymbol{\Sigma}^{-1/2}\hat{\boldsymbol{\mu}}$, $\boldsymbol{\theta} = \sqrt{n}\,\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}^{\text{tgt}}$ and $\mathbf{a} = \sqrt{n}\,\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}^{\text{src}}$. Under the assumption $\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\boldsymbol{\mu}^{\text{tgt}}, \boldsymbol{\Sigma}/n)$, we have $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\theta}, \mathbf{I}_C)$. For any weight λ, possibly depending on $\mathbf{z}$, the whitening is linear, so:

$$n\,\|\boldsymbol{\mu}(\lambda) - \boldsymbol{\mu}^{\text{tgt}}\|^2_{\boldsymbol{\Sigma}^{-1}} = \|\mathbf{a} + \lambda(\mathbf{z} - \mathbf{a}) - \boldsymbol{\theta}\|^2,$$

and the shift score of Equation 6 is $S = \|\mathbf{z} - \mathbf{a}\|^2$.

Risk of the Stein weight. With $\lambda = 1 - (C-2)/S$, the estimator $\mathbf{a} + \lambda(\mathbf{z} - \mathbf{a})$ is the James–Stein estimator with target $\mathbf{a}$, whose risk is $C - (C-2)^2\,\mathbb{E}[1/S]$ by Lemma 1; the positive part $\hat{\lambda}$ of Equation 7 has no larger risk (Efron & Morris, 1973). Since S follows a noncentral χ^2 distribution with C degrees of freedom and noncentrality $\|\boldsymbol{\theta} - \mathbf{a}\|^2 = \Delta$, we have $\mathbb{E}[S] = C + \Delta$. Since the function $x \mapsto 1/x$ is convex on $(0, \infty)$, Jensen's inequality yields $\mathbb{E}[1/S] \geq 1/(C + \Delta)$, which is finite for $C \geq 3$. We conclude:

$$R(\hat{\lambda}) \leq C - \frac{(C-2)^2}{C + \Delta} < C = R(1),$$

where $R(1) = \mathbb{E}\|\mathbf{z} - \boldsymbol{\theta}\|^2 = C$ is the risk of TBN.

Excess over the oracle weight. Subtracting $R(\lambda^\star) = C\Delta/(C + \Delta)$ from Equation 5:

$$R(\hat{\lambda}) - R(\lambda^\star) \leq \frac{C(C + \Delta) - (C-2)^2 - C\Delta}{C + \Delta} = \frac{4(C-1)}{C + \Delta} \leq \frac{4(C-1)}{C} < 4.$$

Therefore, the bound holds for each shift $\boldsymbol{\delta}$ and batch size n, and the excess is largest without any shift ($\Delta = 0$), where the oracle keeps the source mean exactly. □

C. Implementation Details

All experiments run on one CPU machine (15 cores, no GPU) with PyTorch[1] 2.8 and NumPy; the code that produced each number is part of the supplementary material.

Source models. CNN-8 stacks three stages of two 3×3 convolutions with 32, 64 and 128 channels, each followed by BN and ReLU, with max pooling between stages, global average pooling and a linear head. ResNet-14 is the CIFAR residual network of He et al. (2016) with two basic blocks per stage (widths 16, 32, 64), 15 BN layers in total. Table 7 reports the training recipe; training took 11.6 min (CNN-8) and 11.9 min (ResNet-14) of wall-clock time on 7 threads each.

Table 7. Training and evaluation settings.

Setting	Value
Optimizer	SGD, Nesterov momentum 0.9, weight decay 5×10^{-4}
Learning rate	one-cycle schedule, peak 0.1, 15% warm-up
Batch size / epochs	128 / 8 (CNN-8), 5 (ResNet-14)
Augmentation	4-pixel reflect padding + random crop, horizontal flip
Source images for $\boldsymbol{\tau}^2$, $\boldsymbol{\rho}^2$	5,000 random training images, no augmentation
Test images per corruption	1,000 (fixed random subset per corruption)
Floor on the variance	10^{-5}
TENT	Adam, learning rate 10^{-3}, one step per batch, BN affine parameters only

Baselines. TBN uses the biased batch variance over all nHW positions. Prior-BN follows Equation 3 of Schneider et al. (2020) for the mean and the variance. α-BN follows You et al. (2021) and mixes the means and the standard deviations with $\alpha = 0.9$. Oracle-N selects, for each network and batch size, the best $N \in \{2, 4, 8, 16, 32, 64, 128, 256\}$ by the average error over the 15 corruptions; the selected values are $N = 4, 8, 8, 16, 16, 8, 64$ for ResNet-14 and $N = 2, 4, 4, 8, 4, 2, 2$ for CNN-8 at $n = 1, 2, 4, 8, 16, 64, 200$.

Procedure. The network applies Algorithm 1 layer by layer in its forward pass, and each layer receives the features adapted by the layers before it.

Data. CIFAR-10-C (Hendrycks & Dietterich, 2019) was downloaded from its official Zenodo record; we read only the severity-3 and severity-5 slices of the 15 standard corruptions. **Compute.** Besides training, the full evaluation of Table 1 (all rules, ablations and all values of N, both severities) took 7.9 min (ResNet-14, 11 processes) and 12.2 min (CNN-8, 6 processes, partly in parallel), the TENT runs 4.8 and 3.7 min, and the simulation 2 s.

[1] https://pytorch.org

D. Additional Experiments

D.1. Results at Severity 3

Table 8 repeats Table 1 at the milder severity 3. SourceStein has the lowest error from n=2 upward on both networks (TBN is 0.01 points lower at n=64 on CNN-8), while at n=1 the fixed weights of α-BN and Prior-BN, which stay close to the source, are lower; under a mild shift, a single image contains little evidence and the shift bias of the source statistics is small.

Table 8. Error (%) on CIFAR-10-C at severity 3, averaged over 15 corruptions. The best result in each column of a block is **bold**.

Network	Method	1	2	4	8	16	64	200
ResNet-14	Source	35.91	35.91	35.91	35.91	35.91	35.91	35.91
	TBN	89.21	50.52	38.56	31.41	28.12	25.43	24.81
	Prior-BN	33.35	31.03	29.09	27.55	26.87	25.25	24.79
	α-BN	**31.88**	31.04	30.35	29.79	29.50	29.12	29.19
	SourceStein	34.24	**30.47**	**28.21**	**27.04**	**25.97**	**24.96**	**24.57**
CNN-8	Source	32.73	32.73	32.73	32.73	32.73	32.73	32.73
	TBN	89.97	47.99	34.67	27.81	24.47	**21.57**	21.02
	Prior-BN	30.69	28.68	26.51	24.49	23.25	21.71	20.99
	α-BN	**29.53**	28.73	28.43	28.13	27.82	27.70	27.61
	SourceStein	31.30	**28.03**	**25.70**	**23.81**	**22.95**	21.58	**20.91**

D.2. Per-Corruption Results

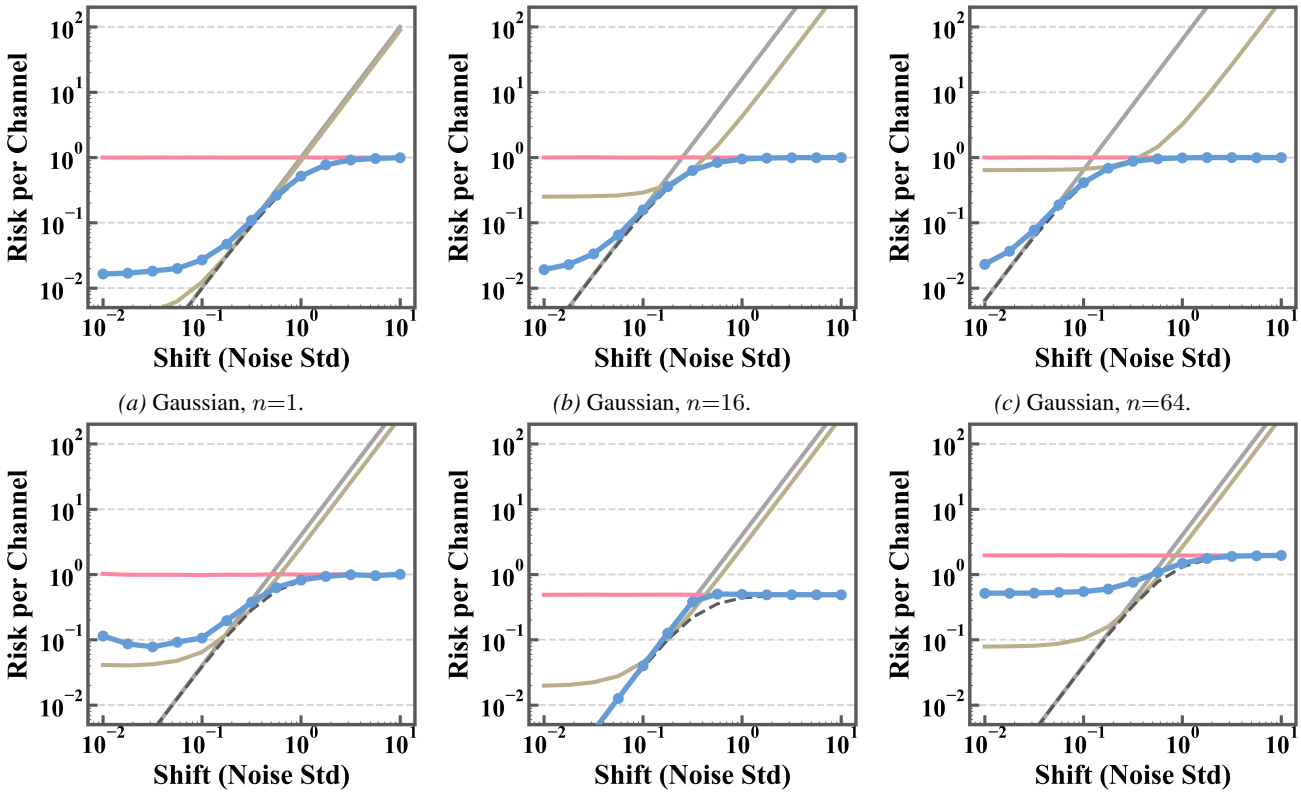
Table 9 breaks down the error at n=4 by corruption. SourceStein is the lowest of the five rules on 12 of the 15 corruptions for each network; its largest gain over the best other rule is on contrast for ResNet-14 (11.6 points) and on Gaussian noise for CNN-8 (9.0 points), and it is not the lowest on elastic transform and JPEG for either network, on glass blur for ResNet-14 and on pixelate for CNN-8.

Table 9. Error (%) per corruption at severity 5 for test batches of n=4 images. The best result in each column of a block is **bold**.

	Method	Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow	Frost	Fog	Brit.	Contr.	Elast.	Pixel.	JPEG
ResNet-14	Source	61.1	58.9	67.0	52.1	65.0	51.7	52.1	43.1	53.4	49.7	24.7	77.4	37.6	57.8	29.3
	TBN	49.0	48.0	52.9	40.3	54.0	39.4	36.3	50.4	45.5	43.4	35.2	43.3	46.2	41.4	41.5
	Prior-BN	44.1	40.2	51.1	34.1	**53.4**	33.9	35.4	40.7	**37.9**	35.3	24.2	46.8	35.3	39.8	28.5
	α-BN	50.2	45.4	57.6	39.3	58.4	39.2	38.4	40.6	42.2	38.5	23.6	60.7	**34.5**	49.0	**28.3**
	SourceStein	**40.4**	**38.3**	**48.2**	**29.7**	54.5	**30.3**	**32.4**	**38.8**	**37.9**	**33.9**	**22.4**	**31.7**	36.2	**39.1**	29.7
CNN-8	Source	65.3	58.8	67.6	52.0	61.8	44.2	46.2	42.3	51.3	44.5	19.8	78.3	33.3	57.4	27.3
	TBN	48.6	46.1	53.7	35.4	53.4	36.8	34.6	44.3	42.4	39.8	29.6	36.9	39.7	**38.8**	39.9
	Prior-BN	48.4	43.2	54.0	33.6	52.9	31.4	34.1	36.8	37.6	33.8	17.0	64.9	**29.9**	42.3	27.8
	α-BN	54.3	49.9	59.9	40.6	57.0	35.5	38.7	37.5	41.5	35.9	18.1	71.5	30.6	50.1	**27.0**
	SourceStein	**39.4**	**35.5**	**50.2**	**29.2**	**52.1**	**28.6**	**31.0**	**35.2**	**36.7**	**29.9**	**16.6**	**35.6**	30.5	41.7	27.5

E. Simulation Study

The simulation draws C=64 per-sample means with unit noise per channel, shifts the target mean from the source mean in a random direction of a given size, and averages the risk of Equation 4 (divided by C) over 4,000 batches per setting; SourceStein uses the source noise level, as in the networks. Figure 8 shows three batch sizes under the Gaussian model and three departures from it at n=4. With Student-t noise of three degrees of freedom, SourceStein remains below TBN at each shift and exceeds the oracle by at most 0.114 per channel. When the true noise is 0.7 times the stored level, SourceStein shrinks too much and exceeds TBN at 6 of the 14 shift sizes; when it is 1.4 times larger, SourceStein stays below TBN but keeps a floor of about 0.5 at zero shift. These cases mark the assumption of Proposition 1 that matters in practice: the noise level must be measured on data whose between-sample spread resembles that of the test data.



(a) Gaussian, n=1. *(b)* Gaussian, n=16. *(c)* Gaussian, n=64.

(d) Student-t_3, n=4. *(e)* Noise 0.7×, n=4. *(f)* Noise 1.4×, n=4.

Figure 8. Risk per channel as a function of the per-channel shift in the simulation (log scales), with the legend of Figure 1: Source (grey), TBN (pink), Prior-BN with N=16 (khaki), the oracle weight (dashed) and SourceStein (blue).