Uncovering Adversarial Risks of Test-Time Adaptation

Tong Wu [1] **Feiran Jia** [2] **Xiangyu Qi** [1] **Jiachen T. Wang** [1]
Vikash Sehwag [1] **Saeed Mahloujifar** [1] **Prateek Mittal** [1]

Abstract

Recently, test-time adaptation (TTA) has been proposed as a promising solution for addressing distribution shifts. It allows a base model to adapt to an unforeseen distribution during inference by leveraging the information from the batch of (unlabeled) test data. However, we uncover a novel security vulnerability of TTA based on the insight that predictions on benign samples can be impacted by malicious samples in the same batch. To exploit this vulnerability, we propose ***D**istribution **I**nvading **A**ttack* (DIA), which injects a small fraction of malicious data into the test batch. DIA causes models using TTA to misclassify benign and unperturbed test data, providing an entirely new capability for adversaries that is infeasible in canonical machine learning pipelines. Through comprehensive evaluations, we demonstrate the high effectiveness of our attack on multiple benchmarks across six TTA methods. In response, we investigate two countermeasures to robustify the existing insecure TTA implementations, following the principle of "security by design". Together, we hope our findings can make the community aware of the *utility-security tradeoffs* in deploying TTA and provide valuable insights for developing robust TTA approaches.

1. Introduction

Test-time adaptation (TTA) (Wang et al., 2021b; Schneider et al., 2020; Goyal et al., 2022) is a cutting-edge machine learning (ML) approach that addresses the problem of distribution shifts in test data (Hendrycks & Dietterich, 2019). Unlike conventional ML methods that rely on a fixed base model, TTA generates batch-specific models to handle different test data distributions. Specifically, when test data are processed batch-wise (bat, 2021), TTA first leverages them to update the base model and then makes the final predictions using the updated model. Methodologically, TTA differs from the conventional ML (*inductive learning*) and falls within the *transductive learning paradigm* (Vapnik, 1998). TTA usually outperforms conventional ML under distribution shifts since 1) it gains the distribution knowledge from the test batch and 2) it can adjust the model adaptively. Empirically, TTA has been shown to be effective in a range of tasks, including image classification (Wang et al., 2021b), object detection (Sinha et al., 2023), and visual document understanding (Ebrahimi et al., 2022).

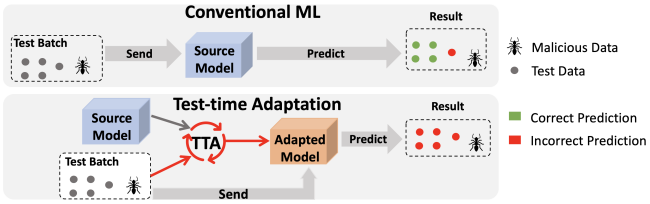


Figure 1. Overview of conventional machine learning and test-time adaptation. Test-time adaptation adapts to the test batch, but is vulnerable to malicious data at test time (in contrast to conventional machine learning).

However, in this work, we highlight **a potential security vulnerability in the test-time adaptation process — an adversary can introduce malicious behaviors into the model by crafting samples in the test batch**. Our key insight is that TTA generates the final predictive model based on the entire test batch rather than making independent predictions for each data as in a conventional ML pipeline. Therefore, the prediction for one entry in a batch will be influenced by other entries in the same batch. As a result, an adversary may submit malicious data at test time to interfere with the generation of the final predictive model, consequently disrupting predictions on other unperturbed data submitted by benign users. This emphasizes the necessity of considering the *utility-security tradeoffs* associated with deploying TTA.

Our Contributions. To exploit this vulnerability, we present a novel attack called ***Distribution Invading Attack (DIA)***, which exploits TTA by introducing malicious data

[1]Princeton University [2]Penn State University. Correspondence to: Tong Wu <tongwu@princeton.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

(Section 4). Specifically, DIA crafts (or uploads) a small number of malicious samples (e.g., 5% of the batch) to the test batch, aiming to induce the mispredictions of benign samples. We formulate DIA as a bilevel optimization problem with outer optimization on crafting malicious data and inner optimization on TTA updates. Next, we transform it into a single-level optimization via approximating model parameters, which can be solved by a projected gradient descent mechanism. DIA is a generic framework that can achieve multiple adversarial goals, including 1) flipping the prediction of a crucial sample to a selected label (targeted attack), 2) degrading performance on all benign data (indiscriminate attack), and achieving the first objective while keeping benign data accuracy (stealthy targeted attack).

We empirically illustrate that DIA achieves a high attack success rate (ASR) on various benchmarks, including CIFAR-10-C, CIFAR-100-C, and ImageNet-C (Hendrycks & Dietterich, 2019) against a range of TTA methods, such as **TeBN** (Nado et al., 2020), **TENT** (Wang et al., 2021b), and **Hard PL** (Lee et al., 2013) in Section 5. Notably, we demonstrate that targeted attacks using 5% of malicious samples in the test batch can achieve over 92% ASR on ImageNet-C. Our evaluation also indicates that DIA performs well across multiple model architectures and data augmentations. Furthermore, the attack is still effective even when there is a requirement for the malicious inputs to be camouflaged in order to bypass the manual inspection.

In response, we explore countermeasures to strengthen the current TTA methods by incorporating the principle of "security by design" (Section 6). Given that adversarially trained models (Madry et al., 2018) are more resistant to perturbations, we investigate the possibility of defending against DIA using them as the base model. In addition, since the vulnerabilities of TTA primarily stem from the insecure computation of Batch Norm (BN), we explore two methods to robustly estimate it. First, we leverage the BN computed during training time, which is robust to DIA, as a prior for the final BN statistics computation. Second, we observe that DIA impacts later BN statistics more than other layers and develop a method to adjust BN statistics accordingly. Our evaluation shows the effectiveness of combining adversarial models and robust BN estimation against DIA, providing guidance to future works for improving TTA robustness.

Overall, our work shows that in effort to enhance utility, using test-time adaptation (transductive learning paradigm) inadvertently amplifies the security risks. Such utility-secruity tradeoff has been deeply explored in inductive learning (Chakraborty et al., 2018), and we urge the community to build upon the prior works in inductive learning, by taking these security risks into account when enhancing the utility with transductive learning. The code is available at `https://github.com/inspire-group/tta_risk`

2. Background and Related Work

In this section, we introduce test-time adaptation and then review the related works in adversarial machine learning. More details can be found in A.

2.1. Notation and Test-Time Adaptation

Let $\mathcal{D}^s := \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N_s}$ be the training data set from the source domain, and $\mathbf{X}^t := \{\mathbf{x}_i^t\}_{i=1}^{N_t}$ be the *unlabeled* test data set from the target domain, where N_s and N_t denote the number of points in $\mathcal{D}^s$ and $\mathbf{X}^t$, respectively. The goal is to learn a prediction rule $f(\cdot;\theta)$ parameterized by θ that can correctly predict the label y_i^t for each $\mathbf{x}_i^t \in \mathbf{X}^t$. In the conventional ML (*inductive learning*) setting, we find the best model parameters θ^s by training on the source dataset $\mathcal{D}^s$. However, in practice, training and test data distribution may shift, and a fixed model $f(\cdot;\theta^s)$ will result in poor performance (Hendrycks & Dietterich, 2019). To address this issue, test-time adaptation (TTA), under *transductive learning* setting, has been proposed (Wang et al., 2021b). Specifically, TTA first obtains a $f(\cdot;\theta^s)$ learned from the source training set $\mathcal{D}^s$ or downloaded from an online source and then adapts to test data $\mathbf{X}^t$ during inference. In this case, TTA can characterize the distribution of $\mathbf{X}^t$, thereby boosting the performance.

2.2. Leveraging the Test Batch in TTA

TTA techniques are typically used when the test data is processed batch by batch in an online manner.[1] Let $\mathbf{X}_B^t = \{\mathbf{x}_i^t\}_{i=1}^{N_{Bt}} \subseteq \mathbf{X}^t$ be a subset (i.e., *batch*) of test data with size N_{Bt}, and $f(\cdot;\theta^{\text{pre}})$ be the pre-adapted model which is going to perform adaption immediately. Initially, we set $\theta^{\text{pre}} = \theta^s$. At each iteration, a new batch of test data $\mathbf{X}_B^t$ is available; parts of the model parameter are updated accordingly, and the final predictions are based on the updated model. Here we introduce two mainstream techniques:

Test-time Batch Normalization (TeBN). Batch Norm (BN) (Ioffe & Szegedy, 2015) normalizes the input features by batch mean and variance, reducing the internal covariate shifts in DNN. Nado et al. (2020) replace the normalization statistics $\{\mu, \sigma^2\}$ on the training set by the statistics of the test batch $\mathbf{X}_B^t$, denoted as $\theta_{\mathcal{B}}(\mathbf{X}_B^t) := \{\mu(\mathbf{X}_B^t), \sigma^2(\mathbf{X}_B^t)\}$, where $\theta_{\mathcal{B}}$ is BN statistics. This can help the model generalize better to the unseen and shifted data distribution.

Self-Learning (SL). SL updates part of model parameters $\theta_{\mathcal{A}}$ by the gradient of loss function $\mathcal{L}_{\text{TTA}}$. Some methods

[1] We exclude single-sample TTA methods (e.g., Zhang et al. (2021a)) which usually perform worse than batched version. In addition, they make independent predictions for each test data, avoiding the risks we discuss in this paper.

like **TENT** (Wang et al., 2021b) minimize the entropy of prediction distribution on a batch of test samples, where loss is $\mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t)$. We then denote the remaining parameters as $\theta_{\mathcal{F}} := \theta^s \setminus (\theta_{\mathcal{A}} \cup \theta_{\mathcal{B}})$, which stay fixed. In other methods like pseudo-labeling (PL), the loss can be formulated as $\mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t, \widetilde{\mathbf{y}})$, where $\widetilde{\mathbf{y}} = \{\widetilde{y}_i\}_{i=1}^{N_{Bt}}$ denote the pseudo-labels. In the standard PL methods, the pseudo-labels $\widetilde{\mathbf{y}}$ can be directly predicted by the teacher, known as **Hard PL** (Lee et al., 2013), or by predicting the class probabilities, known as **Soft PL** (Lee et al., 2013). Later, **Robust PL** proposed by Rusak et al. (2022) and **Conjugate PL** developed by Goyal et al. (2022) further improve the pseudo-labels with more advanced TTA loss. Notably, all **SL** methods usually achieve the best performance when $\theta_{\mathcal{A}}$ equals the affine transformations of BN layers (Rusak et al., 2022).

2.3. Related Work

Previous work on conventional ML risks has focused on adversarial attacks, such as evasion attacks (Biggio et al., 2013; Goodfellow et al., 2015; Carlini & Wagner, 2017), which perturb test data to cause mispredictions by the model during inference. However, our attack on TTA differs in that the malicious samples we construct can target benign and unperturbed data. Another form of ML risk is data poisoning (Biggio et al., 2012; Koh & Liang, 2017; Gu et al., 2017), which involves injecting malicious data into training samples, causing models trained on them to make incorrect predictions. Our attack differs in that we only assume access to unlabeled test data, making it easier to deploy in real-world environments.

The security community has also explored the use of transductive learning to enhance the conventional ℓ_p robustness of ML (Goldwasser et al., 2020; Wu et al., 2020b; Wang et al., 2021a). However, the latter work by Chen et al. (2022b); Croce et al. (2022) demonstrated these defenses through transductive learning have only a small improvement in robustness. Our paper focuses on another perspective, where transductive learning induces new vulnerabilities such that benign and unperturbed data may be misclassified.

Other related works are discussed in Appendix A.5.

3. Threat Model

As no previous literature has studied the vulnerabilities of TTA, we start by discussing the adversary's objective, capabilities, and knowledge for our attack. We consider a scenario in which a victim gets a source model, either trained on source data or obtained online, and seeks to improve its performance using TTA on a batch of test data.

Adversary's Objective. The objective of ***Distribution Invading Attack (DIA)*** is to interfere with the performance of the post-adapted model in one of the following ways: (1) **targeted attack**: misclassifying a crucial targeted sample as a specific label, (2) **indiscriminate attack**: increasing the overall error rate on benign data in the same batch, or (3) **stealthy targeted attack**: achieving targeted attack while maintaining accuracy on other benign samples.

Adversary's Capabilities and Additional Constraints. The attacker can craft and upload a limited number of malicious samples to the test batch during inference. Since the DIA does not make any perturbations on the targeted samples, in our main evaluation, we do not require a constraint for malicious data as long as it is a valid image. However, bypassing the manual inspection is sometimes worthwhile; the adversary may also construct camouflaged malicious samples. Concretely, we consider two constraints on attack samples: (1) ℓ_∞ attacks (Goodfellow et al., 2015), the most common practice in the literature; (2) adversarially generated corruptions (e.g., snow) (Kang et al., 2019), which simulates the target distribution.

Adversary's Knowledge. We consider a white-box setting where the DIA adversary knows the pre-adapted model parameters θ^{pre} and has read-only access to benign samples in the test batch (e.g., a malicious insider is involved). [2] However, the adversary has no access to the training data or the training process. Furthermore, our main attacking methods (used in most experiments) do not require the knowledge of which TTA methods the victim will deploy.

4. Distribution Invading Attack

In this section, we first identify the detailed vulnerabilities (Section 4.1), then formulate Distribution Invading Attack as a general bilevel optimization problem (Section 4.2), and finally discuss constructing malicious samples (Section 4.3).

4.1. Indentifing the Vulnerabilities of TTA

The risk of TTA stems from its transductive learning paradigm, where the predictions on test samples are no longer independent of each other. In this section, we detail how specific approaches used in TTA expose security vulnerabilities.

Re-estimating Batch Normalization Statistics. Most existing TTA methods (Nado et al., 2020; Wang et al., 2021b; Rusak et al., 2022) adopt test-time Batch Normalization (**TeBN**) as a fundamental approach for mitigating distribution shifts when the source model is CNN with BN layers. We denote the input of lth BN layer by $\mathbf{z}_l$, and test-time BN statistics for each BN layer can be computed by $\{\mu \leftarrow \mathbb{E}[\mathbf{z}_l], \sigma^2 \leftarrow \mathbb{E}[(\mu - \mathbf{z}_l)^2]\}$ through the forward path in turn, where the expectation (i.e., average) $\mathbb{E}$ is applied channel-wise on the input. When

[2] We also consider the setting where attackers cannot access benign data in Section 5.5.

calculating BN statistics $\theta_{\mathcal{B}}$ over a batch of test data, any perturbations to some samples in the batch can affect the statistics μ, σ^2 layer by layer and potentially alter the outputs (i.e., $\mathbf{z}_l^{\text{out}} := (\mathbf{z}_l - \mu)/\sigma$) of the other samples in the batch. Hence, adversaries can leverage this property to design Distribution Invading Attacks.

Parameters Update. To further adapt the model to a target distribution, existing methods often update part of model parameters $\theta_{\mathcal{A}}$ by minimizing unsupervised objectives defined on test samples (Wang et al., 2021b; Rusak et al., 2022; Goyal et al., 2022). The updated parameter can be computed by: $\theta^*_{\mathcal{A}} = \arg\min_{\theta_{\mathcal{A}}} \mathcal{L}_{\text{TTA}}(\mathbf{X}^t_B; \theta_{\mathcal{A}})$ Hence, malicious samples inside $\mathbf{X}^t_B$ may perturb the model parameters $\theta^*_{\mathcal{A}}$, leading to incorrect predictions later.

4.2. Formulating DIA as a Bilevel Optimization Problem

We formulate the DIA as an optimization problem to exploit both vulnerabilities mentioned above. Intuitively, we want to craft some malicious samples $\mathbf{X}^t_{mal} := \{\mathbf{x}^t_{mal,i}\}_{i=1}^{N_m}$ to achieve an attack objective (e.g., misclassifying a targeted sample). Then, the test batch $\mathbf{X}^t_B$ comprises $\mathbf{X}^t_{mal}$ and other benign samples $\mathbf{X}^t_{B\setminus mal}$. The pre-adapted model parameter θ^{pre} is composed of parameters $\theta_{\mathcal{A}}$ that will update, BN statistics $\theta_{\mathcal{B}}$, and other fixed parameters $\theta_{\mathcal{F}}$ (i.e., $\theta^{\text{pre}} := \theta_{\mathcal{A}} \cup \theta_{\mathcal{B}} \cup \theta_{\mathcal{F}}$). Here, we use $\mathbb{L}(f(\cdot\,; \theta^*(\mathbf{X}^t_B)))$ to denote the general adversarial loss, and the problem for the attacker can be formulated as the following bilevel optimization:

$$\min_{\mathbf{X}^t_{mal}} \mathbb{L}(f(\cdot\,; \theta^*(\mathbf{X}^t_B))) \tag{1}$$

$$\begin{aligned} s.t.\ \mathbf{X}^t_B &= \mathbf{X}^t_{mal} \cup \mathbf{X}^t_{B\setminus mal}; \quad \theta'_{\mathcal{B}} = \{\mu(\mathbf{X}^t_B), \sigma^2(\mathbf{X}^t_B)\}; \\ \theta^*_{\mathcal{A}} &= \arg\min_{\theta_{\mathcal{A}}} \mathcal{L}_{\text{TTA}}(\mathbf{X}^t_B; \theta_{\mathcal{A}}, \theta'_{\mathcal{B}}, \theta_{\mathcal{F}}); \\ \theta^*(\mathbf{X}^t_B) &= \theta^*_{\mathcal{A}} \cup \theta'_{\mathcal{B}} \cup \theta_{\mathcal{F}}; \end{aligned} \tag{2}$$

where $\theta'_{\mathcal{B}}$ is the updated BN statistics given the test batch data, $\theta^*_{\mathcal{A}}$ is the parameter that is optimized over TTA loss, and $\theta^*(\mathbf{X}^t_B)$ is the optimized model parameters containing $\theta^*_{\mathcal{A}}$, $\theta'_{\mathcal{B}}$, and other fixed parameters $\theta_{\mathcal{F}}$. In most cases, test-time adaptation methods perform a single-step gradient descent update (Wang et al., 2021b), and the inner optimization for TTA simplifies to $\theta^*_{\mathcal{A}} = \theta'_{\mathcal{A}} = \theta_{\mathcal{A}} - \partial\mathcal{L}_{\text{TTA}}(\mathbf{X}^t_B)/\partial\theta_{\mathcal{A}}$. Now, we discuss how we design the specific adversarial loss $\mathbb{L}(f(\cdot\,; \theta^*(\mathbf{X}^t_B)))$ for achieving various objectives.

Targeted Attack. We aim to cause the model to predict a specific incorrect targeted label $\dot{y}_{tgt}$ for a targeted sample $\mathbf{x}^t_{tgt} \in \mathbf{X}^t_{B\setminus mal}$. Thus, the objective can be formulated as:

$$\widehat{\mathbf{X}}^t_{mal} := \arg\min_{\mathbf{X}^t_{mal}} \mathcal{L}(f(\mathbf{x}^t_{tgt}; \theta^*(\mathbf{X}^t_B)), \dot{y}_{tgt}) \tag{3}$$

where $\mathcal{L}$ is the cross-entropy loss.

Indiscriminate Attack. The objective turns to degrade the performance of all benign samples as much as possible.

Algorithm 1 for constructing Distribution Invading Attack

Input: Pre-adapted model parameters $\theta^{\text{pre}} = \theta_{\mathcal{A}} \cup \theta_{\mathcal{B}} \cup \theta_{\mathcal{F}}$, test batch $(\mathbf{X}^t_B; \mathbf{y}^t_B)$ which contains malicious samples $\mathbf{X}^t_{mal}$ and benign samples $\mathbf{X}^t_{B\setminus mal}$, targeted samples $\mathbf{x}^t_{tgt}$ and incorrect targeted label $\dot{y}_{tgt}$, attack learning rates α, constraint ϵ, number of steps N, TTA update rate: η, perturbation $\boldsymbol{\delta_m}$=**0**
Output: Perturbed malicious input $\mathbf{X}^t_{mal} + \boldsymbol{\delta_m}$
for step $= 1, 2, \ldots, N$ **do**:
  $\mathbf{X}^t_B \leftarrow (\mathbf{X}^t_{mal} + \boldsymbol{\delta_m}) \cup \mathbf{X}^t_{B\setminus mal}$
  $\theta'_{\mathcal{B}} \leftarrow \{\mu(\mathbf{X}^t_B), \sigma^2(\mathbf{X}^t_B)\}$
  (Optional) $\theta'_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} - \eta \cdot \partial\mathcal{L}_{\text{TTA}}(\mathbf{X}^t_B)/\partial\theta_{\mathcal{A}}$
  # $\theta'_{\mathcal{A}} \approx \theta_{\mathcal{A}}$ in the single-level version.
  $\theta^* \leftarrow \theta'_{\mathcal{A}} \cup \theta'_{\mathcal{B}} \cup \theta_{\mathcal{F}}$
  $\boldsymbol{\delta_m} \leftarrow \Pi_\epsilon\,(\boldsymbol{\delta_m} - \alpha \cdot \text{sign}(\nabla_{\boldsymbol{\delta_m}} \mathbb{L}(f(\cdot\,; \theta^*(\mathbf{X}^t_B)))))$
  # $\mathbb{L}$ is chosen from Eq. (3), Eq. (4), or Eq. (5)
end for
return $\widehat{\mathbf{X}}^t_{mal} = \mathbf{X}^t_{mal} + \boldsymbol{\delta_m}$

Given the correct labels of benign samples $\mathbf{y}^t_{B\setminus mal}$, we define the goal of indiscriminate attack as follows:

$$\widehat{\mathbf{X}}^t_{mal} := \arg\min_{\mathbf{x}^t_{mal}} -\mathcal{L}(f(\mathbf{X}^t_{B\setminus mal}; \theta^*(\mathbf{X}^t_B)), \mathbf{y}^t_{B\setminus mal}). \tag{4}$$

Stealthy Targeted Attack. In some cases, when performing **targeted attack**, the performance of the other benign samples $\mathbf{X}^t_{B\setminus(tgt\cup mal)}$, which is the whole test batch excluding malicious and targeted data, may drop. A solution is conducting targeted attacks and maintaining the accuracy of other benign data simultaneously, which is:

$$\begin{aligned} \widehat{\mathbf{X}}^t_{mal} := \arg\min_{\mathbf{x}^t_{mal}}\ & \mathcal{L}(f(\mathbf{x}^t_{tgt}; \theta^*(\mathbf{X}^t_B)), \dot{y}_{tgt}) + \\ & \omega * \mathcal{L}(f(\mathbf{X}^t_{B\setminus(tgt\cup mal)}; \theta^*(\mathbf{X}^t_B)), \mathbf{y}^t_{B\setminus(tgt\cup mal)}). \end{aligned} \tag{5}$$

We introduce a new weight term ω to capture the trade-off between these two objectives.

4.3. Constructing Malicious Inputs via Projected Gradient Descent

We solve the bilevel optimization problems defined in the last section via iterative *projected gradient descent*, summarized in Algorithm 1. Our solution generalizes across the three adversary's objectives, where $\mathbb{L}(f(\cdot\,; \theta^*(\mathbf{X}^t_B)))$ can be replaced by Eq. (3), Eq. (4), or Eq. (5). We follow the general projected gradient descent method but involve TTA methods. Concretely, Line 4 updates the test batch, and Line 5 computes the test-time BN. Then, Line 6 and Line 7 perform parameter updates. In Line 8, we compute the gradient toward the objective (three attacks we discussed previously) and update the perturbation $\boldsymbol{\delta_m}$ with α learning rate. Since, in most cases, we do not consider a constraint for the malicious images, the projection Π_ϵ is to ensure the

images are valid in the [0,1] range. For ℓ_∞ constrained DIA attacks, we will also leverage the Π_ϵ to clip $\boldsymbol{\delta_m}$ with constraint ϵ (e.g., $\epsilon = 8/255$). Finally, we get the optimal malicious output samples $\widehat{\mathbf{X}}^t_{mal} = \mathbf{X}^t_{mal} + \boldsymbol{\delta_m}$.

Implementation. While Algorithm 1 is intuitive, the gradient computation can be inconsistent and incur a high computational cost due to the bilevel optimization. Furthermore, we notice that TTA methods only perform a short and one-step update on $\theta_{\mathcal{A}}$ parameters at each iteration (Line 6). Therefore, we approximate $\theta'_{\mathcal{A}} \approx \theta_{\mathcal{A}}$ in most experiments, which makes the inner TTA gradient update (Line 6) optional.[3] Since the re-estimated $\theta'_{\mathcal{B}}$ does not involve any inner gradient computation, we then decipher the problem as a single-level optimization. Our empirical evaluation indicates the effectiveness of our method, reaching comparable or even higher results (presented in Appendix C.2). An additional benefit of our method is we no longer need to know which TTA methods the victim will choose. We also provide some theoretical analysis in Appendix B.1

5. Evaluation of Distribution Invading Attacks

In this section, we report the results of Distribution Invading Attacks. We present the experimental setup in Section 5.1, discuss the main results for targeted attacks in Section 5.2, present the results for indiscriminate and stealthy targeted attacks in Section 5.3, consider extra constraints in Section 5.4, and demonstrate the attack effectiveness under relaxed assumptions in Section 5.5. Further results, including more ablation studies, are presented in Appendix D.

5.1. Experimental Setup

Dataset & Architectures. We evaluate our attacks on well-established distribution shift benchmarks, namely CIFAR-10 to CIFAR-10-C, CIFAR-100 to CIFAR-100-C, and ImageNet to ImageNet-C (Hendrycks & Dietterich, 2019), where the severity of the corruption is set to medium (level 3). We primarily use ResNet-26 (He et al., 2016) for CIFAR-C[4], and ResNet-50 for ImageNet-C.

Test-time Adaptation Methods. We select six TTA methods, which are **TeBN** (Nado et al., 2020), **TENT** (Wang et al., 2021b), **Hard PL** (Lee et al., 2013), **Soft PL** (Lee et al., 2013), **Robust PL** (Rusak et al., 2022), and **Conjugate PL** (Goyal et al., 2022). We follow the settings of their experiments, where the batch size is **200**, and all other hyperparameters are also default values. As a result, all TTA methods significantly boost the **corruption accuracy** (i.e., the accuracy on benign corrupted test data like ImageNet-C), which is shown in Appendix C.3.

[3]Line 6 is necessary for models without BN layers.

[4]CIFAR-C denotes both CIFAR-10-C and CIFAR-100-C.

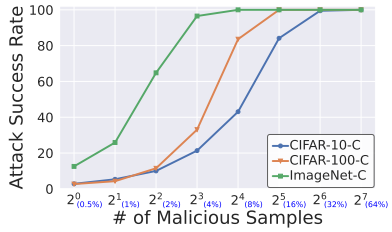


Figure 2. Success rate of our proposed attack across numbers of malicious samples from 1 to 128 (0.5% to 64%). [TTA: **TeBN**]

Attack Settings (Targeted Attack). We consider each test batch as an *individual* attacking trial where we randomly pick one targeted sample with a targeted label.[5] The attacker can inject a small set of malicious data inside this batch without restrictions as long as their pixel values are valid. In total, there are 750 trials for CIFAR-C and 375 trials for ImageNet-C. We estimate our attack effectiveness by **attack success rate (ASR)** averaged across all trials. Note that except **TeBN**, for each trial, TTA methods use the current batch to update $\theta_{\mathcal{A}}$, and send it to the next trial. Hence, θ^{pre} is different for different trials. Further experimental setup details are in Appendix D.1.

5.2. Main Results of Targeted Attack

Distribution Invading Attack achieves a high attack success rate (ASR) across benchmarks and TTA methods (Table 1). We select $N_m = \{10, 20, 40\}$ of malicious samples out of 200 data in a batch for CIFAR-C and $N_m = \{5, 10, 20\}$ for ImageNet-C. By constructing 40 malicious samples on CIFAR-10-C and CIFAR-100-C, our proposed attack can shift the predictions of the victim sample to a random targeted label in more than 82.93% trials. For ImageNet-C, just 10 malicious samples are sufficient to attack all TTA methods with an attack success rate of more than 92.80%. The ASR of **TeBN** are higher than other TTA methods since the attacks here only exploited the vulnerabilities from re-estimating batch normalization statistics. We demonstrate that further parameter updates of other TTA methods cannot greatly alleviate the attack effectiveness. For example, the drop of ASR is less than 6.7% for ImageNet-C with 10 malicious samples.

DIA reaches near-100% ASR, with 64, 32, and 16 malicious samples for CIFAR-10-C, CIFAR-100-C, and ImageNet-C, respectively (Figure 2). We then evaluate our attack with more comprehensive experiments across the number of malicious samples. Specifically, 1 to 128 malicious samples are considered, which is 0.5% to 64% of batch size. **TeBN** is selected as the illustrated TTA method (Appendix D.2 presents more methods). Generally, the attack success rate increases when the attacker can control more samples. For CIFAR-10-C, we observe that

[5]Unless otherwise specified, most experiments in this paper are conducted using targeted attack.

Table 1. Attack success rate of Distribution Invading Attack (targeted attacks) across benchmarks and TTA methods. N_m refers to the number of malicious data, where the batch size is **200**.

Dataset	N_m	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
CIFAR-10-C (ResNet26)	10 (5%)	25.87	23.20	25.33	23.20	24.80	23.60
	20 (10%)	55.47	45.73	48.13	47.47	49.47	45.73
	40 (20%)	92.80	83.87	84.27	82.93	86.93	85.47
CIFAR-100-C (ResNet26)	10 (5%)	46.80	26.40	31.20	27.60	32.13	26.13
	20 (10%)	93.73	72.80	87.33	78.53	82.93	71.60
	40 (20%)	100.00	100.00	99.87	100.00	99.87	100.00
ImageNet-C (ResNet50)	5 (2.5%)	80.80	75.73	69.87	62.67	66.40	57.87
	10 (5%)	99.47	98.67	96.53	94.13	96.00	92.80
	20 (10%)	100.00	100.00	100.00	100.00	100.00	100.00

Table 2. Effectiveness of the Distribution Invading Attack across various model architectures and data augmentations. [CIAFR-C: N_m=40; ImageNet-C: N_m=10]. Table 7 presents full results.

Dataset	Architectures	TeBN(%)	TENT(%)	Hard PL(%)
CIFAR-10-C	ResNet-26	92.80	83.87	84.27
	VGG-19	79.07	65.33	67.47
	WRN-28	93.73	89.60	90.80
CIFAR-100-C	ResNet-26	100.00	100.00	99.87
	VGG-19	88.00	82.93	86.53
	WRN-28	97.47	83.60	87.07
Dataset	**Augmentations**	**TeBN(%)**	**TENT(%)**	**Hard PL(%)**
ImageNet-C	Standard	99.47	98.67	96.53
	AugMix	98.40	96.00	93.60
	DeepAugment	96.00	94.67	91.20

Table 3. Average benign corruption error rate of TTA when deploying DIA indiscriminate attack. Table 8 presents full results.

Dataset	N_m	TeBN(%)	TENT(%)	Hard PL(%)
CIFAR-10-C (ResNet-26)	0 (0%)	10.73	10.47	11.05
	40 (20%)	28.02	27.01	27.91
CIFAR-100-C (ResNet-26)	0 (0%)	34.52	33.31	34.66
	40 (20%)	58.41	54.44	56.59
ImageNet-C (ResNet-50)	0 (0%)	47.79	45.45	43.42
	20 (10%)	79.03	75.01	70.26

the attacker has to manipulate $\sim$64 (32% of the batch size) samples to ensure victim data will be predicted as a targeted label with a near-100% chance. Although 32% seems to be a relatively strong assumption compared to conventional poisoning attacks, it is worth noting that DIA only requires perturbing test data which might not be actively monitored.

DIA also works across model architectures and data augmentations (Table 2). Instead of ResNet-26, we also consider two more common architectures: VGG (Simonyan & Zisserman, 2015) with 19 layers (VGG-19), and Wide ResNet (Zagoruyko & Komodakis, 2016) with a depth of 28 and a width of 10 (WRN-28) on CIFAR-C dataset. Our proposed attack is also effective across different architectures, despite the attack success rates suffering some degradations ($<$20%) for VGG-19. We hypothesize that the model with more batch normalization layers is more likely to be exploited by attackers, where ResNet-26, VGG-19, and WRN-28 contain 28, 16, and 25 BN layers, respectively. We further refine the statement through more experiments in Appendix D.3. In addition, we also select two more data augmentation methods, AugMix (Hendrycks et al., 2020) and DeepAugment (Hendrycks et al., 2021), on top of ResNet-50, which is known to improve the model generalization on the corrupted dataset. Strong data augmentation techniques provide mild mitigations against our attacks (with 10 malicious samples) but still mistakenly predict the targeted data in more than 91% of trials.

5.3. Alternating Attacking Objective

Our previous DIA evaluations focus on targeted attacks. However, as we discussed in Section 4.2, malicious actors can also leverage our attacking framework to achieve alternative goals by slightly modifying the loss function.

5.3.1. Indiscriminate Attack

The first alternative objective is to degrade the performance on all benign samples, which is done by leveraging Eq. (4). Here, we adopt the **corruption error rate** on the benign corrupted dataset as the attack evaluation metric.

By injecting a small set of malicious samples, the error rate grows (Table 3). Besides attack effectiveness, we report the error rate for 0 malicious samples (which stands for no attacks) as the standard baseline. Our attack causes the error rate on benign samples to rise from $\sim$11% to $\sim$28% and from $\sim$34% to $\sim$56% for the CIFAR-10-C and CIFAR-100-C benchmarks, respectively. Furthermore, only 20 malicious samples (10%) for ImageNet-C boost the error rate to more than 70%. Since all benign samples remain unperturbed, the increasing error rate demonstrates the extra risks of TTA methods compared to the conventional ML.

5.3.2. Stealthy Targeted Attack

In Table 1, $\sim$40 samples are needed for the CIFAR-C dataset to achieve a high attacking performance. Thus, the malicious effect might also affect the predictions for other benign samples, resulting in losing attacking stealth. We adopt Eq. (5) to simultaneously achieve targeted attacks and maintain corruption accuracy. Here, we use the **corruption**

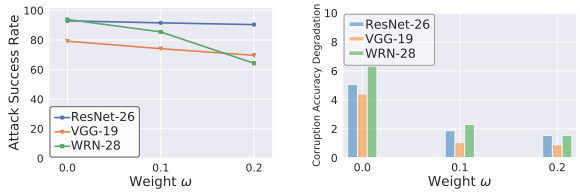
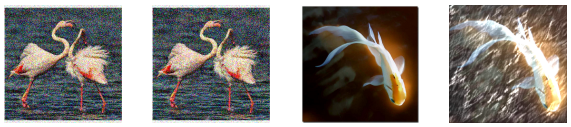


Figure 3. Adjusting the weight ω achieves a high attack success rate (line) and a high benign corruption accuracy (bar). [Benchmark: CIFAR-10-C, N_m=40, TTA method: **TeBN**]



(a) (b) (c) (d)

Figure 4. (a) Test data from ImageNet-C with Gaussian noise. (b) L_∞ constrained DIA on test data (ϵ=$8/255$). (c) A clean data from the ImageNet validation set. (d) Snow attack on the clean data.

accuracy degradation on benign samples to measure the stealthiness.

When $\omega = 0.1$, stealthy DIA can both achieve a high attack success rate and maintain corruption accuracy (Figure 3). We select CIFAR-10-C with 40 malicious data and **TeBN** method, where $\omega = \{0, 0.1, 0.2\}$. We observe that if $\omega = 0.1$, the corruption accuracy degradation drops to $\sim$2%. At the same time, the ASR remains more than 75%. Appendix D.5 has more results on CIFAR-100-C.

5.4. Additional Constraints on Malicious Images

We further consider the stealth of malicious samples to avoid suspicion. The model may reject test samples that are anomalous and refuse to adapt based on them.

5.4.1. ℓ_p Bound

Like much other literature in the adversarial machine learning community, we adopt the ℓ_p constraints as imperceptible metrics to perturb the malicious samples. Specifically, we conduct targeted attack experiments on the ImageNet-C by varying the ℓ_∞ bound of 5 malicious data samples. Figure 5 reports the attacking effectiveness trends in terms of various ℓ_∞ constraints, **where DIA with ϵ = $32/255$ reach similar performance with unconstrained attacks**. Furthermore, we specifically select the $\epsilon = 8/255$ to run extra experiments on the number of malicious samples, as the resulting images are almost imperceptible to the original images (showed in Figure 4(b) and Figure 26). **As a result, DIA achieves near-100% attack success rate, with 32 (ϵ = $8/255$) malicious samples for ImageNet-C.**

5.4.2. Simulated Corruptions

Since our out-of-distribution benchmark is composed of common corruptions, another idea is to leverage such intrinsic properties and generate "imperceptible" adversarial

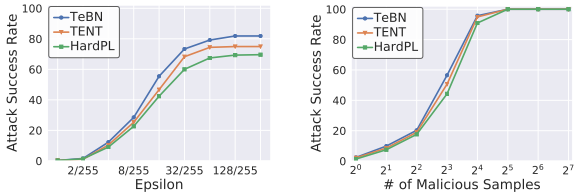


Figure 5. Illustration of the attack success rate across various ϵ of L_∞ constraints [$N_m = 5$] (left) and the different number of malicious samples (right) [$\epsilon = 8/255$]. [Targeted Attack]

Table 4. Average attack success rate of adversarially optimized snow corruptions and fog corruptions. [N_m=20; Targeted Attack]

Dataset	Corruption/Attack	TeBN(%)	TENT(%)	Hard PL(%)
ImageNet-C (ResNet50)	Snow/Snow Attack	64.00	68.00	68.00
	Fog/Fog Attack	84.00	76.00	72.00

samples adaptively. For example, we can apply the adversarially optimized snow distortions to the clean images and insert them into the test data in snow distribution. Then, these injected malicious images (shown in Figure 4(d)) are hard to be distinguished from benign corrupted data. For implementation, we again compute the gradient of the loss function in Eq. (3) and adopt the same approach as Kang et al. (2019).

Adversarially optimized simulated corruption is another effective and input-stealthy DIA vector (Table 4). We apply the Snow attack to the ImageNet-C with snow corruptions, similar to Fog. By inserting 20 malicious samples, the attack success rate reaches at least 60% and 72% for Snow and Fog, respectively. Our findings show attackers can leverage test distribution knowledge to develop a better threat model. More details are presented in Appendix D.6.

5.5. Relaxing the Assumptions of Attacker's Knowledge

In this subsection, we evaluate the performance under relaxed assumptions of DIA attacks, including no access to benign data and randomly selected test batch.

5.5.1. No Access to Benign Samples

We first illustrate the feasibility of our attack method, despite lacking access to benign samples within the test batch. To approximate the benign user data during the test phase, we employ data sourced from the training set. More specifically, we select a random subset of data for each attack iteration, apply analogous noise (i.e., noise similar to the one applied to the targeted sample), and use it as a replacement for benign data. Next, we conduct the targeted attack on the ImageNet-C dataset (Gaussian noise subset), varying the amount of injected malicious data.

Table 5 presents the experimental outcomes when attackers lack access to benign samples at test time. Compared to our

Table 5. Attack success rate with two relaxed assumptions of attacker's capability (i.e., access/no access to benign data and whether the random batch selection (RBS) is considered). [ImageNet; Targeted Attack]. Table 10 presents full results

N_m	Benign Data	RBS	TeBN(%)	TENT(%)	Hard PL(%)
5(2.5%)	Access	✗	92.00	92.00	72.00
5(2.5%)	No access	✗	64.00	52.00	44.00
5(2.5%)	No access	✓	60.00	44.00	40.00
10(5%)	Access	✗	100.0	100.0	100.0
10(5%)	No access	✗	100.0	92.00	84.00
10(5%)	No access	✓	100.0	92.00	84.00

previous scenario (where the attacker has access to benign samples in the test batch), the attack success rate drops ∼30% if 5 malicious data samples are injected but achieve comparable performance if more malicious data samples are injected. This indicates that **our DIA attack is still highly effective even if benign samples cannot be accessed**.

5.5.2. Random Batch Selection

We also consider a more practical setting where the test batch is chosen randomly, meaning that targeted data can be in any batch. The challenge arises from the unpredictability of the number and selection of malicious samples, which complicates managing their effectiveness. To eliminate randomness within the malicious samples, we constrain them to be identical. Then, our goal is twofold: (1) perform the targeted attack on the batch containing the targeted sample, accounting for the uncertainty of the number of malicious samples while ensuring identicalness through the same initialization and updates of malicious samples; and (2) insert a sufficient number of malicious examples into the test set. To ensure the targeted data comprise n% malicious data in expectation, we should inject n% of malicious data into the entire test set.

Taking the random batch selection into account won't largely affect the DIA performance. Our analysis also extends to the targeted attack on the ImageNet-C dataset (Gaussian noise subset) under conditions where the attacker has no prior knowledge of which batch contains the targeted data. In Table 5, we document the results when the attacker has no access to benign samples, and the batch selection is random. Here, the number of malicious data is in expectation. Interestingly, we observe that the attack success rates yield identical performance when the number of malicious data is 10.

6. Mitigating Distribution Invading Attacks

Next, we turn our attention to developing mitigation methods. Our goal is two folds: maintaining the benefits of TTA methods and mitigating Distribution Invading Attacks. Note that we present additional results, including all CIFAR-C experiments, in Appendix E.

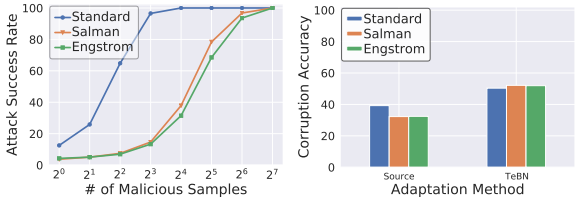


Figure 6. (Left) Attack success rate of DIA against robust models, including Salman et al. (2020) and Engstrom et al. (2019), across numbers of malicious samples. As a reference, 128 is 64% of the whole batch containing 200 samples. (Right) Corruption accuracy of robust models. [TTA method: **TeBN**]

6.1. Leveraging Robust Model to Mitigate DIA

Our first idea is to replace the source model with a robust model (i.e., adversarially trained models (Madry et al., 2018)). Our intuition is that creating adversarial examples during training causes shifts in the batch norm statistics. (Adversarially) training with such samples will robustify the model to be resistant to BN perturbations. We evaluate the targeted DIA against robust ResNet-50 trained by Salman et al. (2020) and Engstrom et al. (2019), which are the best ResNet-50 models from RobustBench (Croce et al., 2020). Since our single-level attacks only exploit the vulnerabilities of re-estimating the BN statistics, evaluating defenses on TTA with updating parameters could give a false sense of robustness. Therefore, most countermeasure experiments mainly focus on the **TeBN** method.

Adversarially trained models boost robustness against DIA and maintain the corruption accuracy (Figure 6). We report the ASR curve of two robust models and observe that they significantly mitigate the vulnerabilities from the test batch. For example, with 8 malicious samples, robust models degrade ASR by ∼80% for ImageNet-C. At the same time, the **TeBN** method significantly improves the corruption accuracy of robust models, reaching even higher results. However, DIA still achieves more than 70% success rate with 32 malicious samples (16% of the whole batch).

6.2. Robust Estimate of Batch Normalization Statistics

We then seek to mitigate the vulnerabilities by robustifying the re-estimation of Batch Norm statistics.

Smoothing via training-time BN statistics. Since training-time BN cannot be perturbed by DIA, we can combine the training-time and test-time BN statistics. This approach is also mentioned in (Schneider et al., 2020; You et al., 2021); however, their motivation is to stabilize the BN estimation when batch size is small (e.g., <32) and improve the corruption accuracy. It can be formulated as $\bar{\mu} = \tau\mu_s + (1-\tau)\mu_t, \quad \bar{\sigma}^2 = \tau\sigma_s^2 + (1-\tau)\sigma_t^2$, where $(\bar{\mu}, \bar{\sigma}^2)$ stand for final BN statistics, (μ_s, σ_s^2) are training-time BN, and (μ_t, σ_t^2) are test-time BN. We view the training-time BN statistics as a robust prior for final estimation and adopt

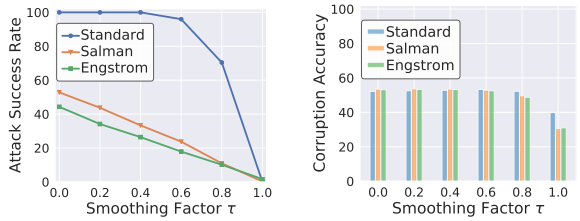


Figure 7. Controlling $\tau = 0.6$ can degrade the attack success rate (line) while maintaining high corruption accuracy (bar). [N_m=20]

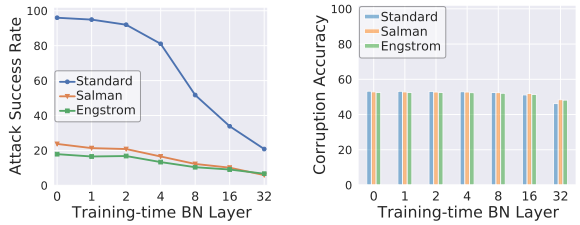


Figure 8. Balancing layer-wise BN can degrade the ASR (line) while maintaining high corruption accuracy (bar). [N_m=20, τ=0.6]

smoothing factor τ to balance the weight.

Leveraging Training-time Batch Norm statistics mitigates the vulnerabilities for both standard and adversarial models (Figure 7). We specifically select $N_m = 20$ and set $\tau = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$, where $\tau = 0.0$ ignores the test-time BN and $\tau = 1.0$ ignores training-time BN. It appears that improving τ generally results in both ASR and corruption accuracy drops on ImageNet-C. However, the degradation in corruption accuracy happens only when $\tau > 0.6$. Therefore, setting $\tau = 0.6$ is a suitable choice, which can mitigate the ASR to $\sim$20% for 20 malicious samples with robust models.

Adptively Selecting Layer-wise BN statistics. We also explore and understand the DIA by visualizing each layer BN in Appendix E.4. Given the discovery where BN statistics shift on the latter layers when applying DIA, we can strategically select training or test time BN statistics for different layers. Hence, we take advantage of training-time BN for the last few layers to constrain the malicious effects.

Adaptive Layer-wise BN further constrains the malicious effect from DIA (Figure 8). Given $\tau = 0.6$ is a suitable choice for whole BN layers, we further leverage full training-time BN ($\tau = 1.0$) for the last $N_{tr} = \{0, 1, 2, 4, 8, 16\}$ BN layers. It appears that increasing training-time BN layers (from $N_{tr} = 0$ to $N_{tr} = 16$) results in tiny corruption accuracy drops ($\sim$2%) and generally helps the robustness against DIA. For example, if we set $N_{tr} = 8$, ASR drops $\sim$40% for the standard model with $N_m = 20$.

In conclusion, applying both approaches, our best results achieve a negligible corruption accuracy degradation and mitigate ASR by $\sim$50% for the standard model and $\sim$40% for the robust model on ImageNet-C. However, our mitigations have not fully resolved the issue, where increasing the number of malicious data samples may still allow successful DIA attacks with a high probability. We encourage future researchers to consider the potential adversarial risks when developing TTA techniques.

7. Discussion and Future Work

While our proposed attacks are promising in terms of effectiveness, several limitations exist. Many recent works in test-time adaptation have been proposed, which leverage different adaptation techniques (e.g., (Niu et al., 2022)). Most of them still suffer from the two vulnerabilities we identified in Section 4 and can be attacked by DIA directly (see Appendix D.12.1 and Appendix D.12.2). However, an adaptive attack design should be considered if the TTA techniques change significantly (e.g., methods neither using test-time BN nor self-learning, see Appendix D.12.3). Furthermore, methods under the general transductive learning settings (i.e., predictions affected by test batch) also involve similar risks, which we encourage the community to explore in future studies.

Another direction is relaxing the adversary's knowledge assumption. When model architectures and parameters are not exposed to adversaries (black-box threat model), launching DIA attacks could become more challenging. Future works should consider methods using model ensemble (Geiping et al., 2020) or diverse inputs (Xie et al., 2018) to boost the transferability of DIA's malicious data.

8. Conclusion

In this work, we investigate adversarial risks of test-time adaptation (TTA). First, we present a novel framework called Distribution Invading Attack, which can be used to achieve new malicious goals. Significantly, we prove that manipulating a small set of test samples can affect the predictions of other benign inputs if adopting TTA, which is not studied at all in previous literature. We then explore mitigation strategies, such as utilizing an adversarially-trained model as a source model and robustly estimating BN statistics. Overall, our findings uncover the risks of TTA and inspire future works build robust and effective TTA techniques.

Acknowledgements. We are grateful to Ahmed Khaled, Chong Xiang, Ashwinee Panda, Tinghao Xie, William Yang, Beining Han, Liang Tong, and Prof. Olga Russakovsky for providing generous help and insightful feedback on our project. This works was supported in part by the National Science Foundation under grant CNS-2131938, the ARL's Army Artificial Intel- ligence Innovation Institute (A2I2), Schmidt DataX award, Princeton E-ffiliates Award, and Princeton Gordon Y. S. Wu Fellowship.

References

Online versus batch prediction. https://cloud.google.com/ai-platform/prediction/docs/online-vs-batch-prediction, 2021.

Bartler, A., Bühler, A., Wiewel, F., Döbler, M., and Yang, B. Mt3: Meta test-time training for self-supervised test-time adaption. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 3080–3090. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/bartler22a.html.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *ECML/PKDD*, 2013.

Carlini, N. Poisoning the unlabeled dataset of semi-supervised learning. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

Carlini, N. and Terzis, A. Poisoning and backdooring contrastive learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=iC4UHbQ01Mp.

Carlini, N. and Wagner, D. A. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. Adversarial attacks and defences: A survey. *ArXiv*, abs/1810.00069, 2018.

Chen, J., Cheng, Y., Gan, Z., Gu, Q., and Liu, J. Efficient robust training via backward smoothing. In *AAAI Conference on Artificial Intelligence*, 2022a.

Chen, J., Wu, X., Guo, Y., Liang, Y., and Jha, S. Towards evaluating the robustness of neural networks learned by transduction. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=_5js_8uTrx1.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Croce, F., Gowal, S., Brunner, T., Shelhamer, E., Hein, M., and Cemgil, A. T. Evaluating the adversarial robustness of adaptive test-time defenses. *ArXiv*, abs/2202.13711, 2022.

Dai, S., Mahloujifar, S., and Mittal, P. Parameterizing activation functions for adversarial robustness. *2022 IEEE Security and Privacy Workshops (SPW)*, pp. 80–87, 2022.

Damodaran, B. B., Kellenberger, B., Flamary, R., Tuia, D., and Courty, N. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 447–463, 2018.

Di, J. Z., Douglas, J., Acharya, J., Kamath, G., and Sekhari, A. Hidden poison: Machine unlearning enables camouflaged poisoning attacks. *ArXiv*, abs/2212.10717, 2022.

Döbler, M., Marsden, R. A., and Yang, B. Robust mean teacher for continual and gradual test-time adaptation. *ArXiv*, abs/2211.13081, 2022.

Ebrahimi, S., Arik, S. O., and Pfister, T. Test-time adaptation for visual document understanding, 2022. URL https://arxiv.org/abs/2206.07240.

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

Fournier, A., Fussell, D. S., and Carpenter, L. C. Computer rendering of stochastic models. *Commun. ACM*, 25:371–384, 1982.

Galstyan, A. G. and Cohen, P. R. Empirical comparison of "hard" and "soft" label propagation for relational classification. In *ILP*, 2007.

Gandelsman, Y., Sun, Y., Chen, X., and Efros, A. A. Test-time training with masked autoencoders. *ArXiv*, abs/2209.07522, 2022.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Gao, J., Zhang, J., Liu, X., Darrell, T., Shelhamer, E., and Wang, D. Back to the source: Diffusion-driven test-time adaptation. *arXiv preprint arXiv:2207.03442*, 2022a.

Gao, Y., Shi, X., Zhu, Y., Wang, H., Tang, Z., Zhou, X., Li, M., and Metaxas, D. N. Visual prompt tuning for test-time domain adaptation. *ArXiv*, abs/2210.04831, 2022b.

Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' brew: Industrial scale data poisoning via gradient matching. *ArXiv*, abs/2009.02276, 2020.

Geiping, J., Fowl, L., Somepalli, G., Goldblum, M., Moeller, M., and Goldstein, T. What doesn't kill you makes you robust (er): Adversarial training against poisons and backdoors. *arXiv preprint arXiv:2102.13624*, 2021.

Goldwasser, S., Kalai, A. T., Kalai, Y., and Montasser, O. Beyond perturbations: Learning guarantees with arbitrary adversarial test examples. *Advances in Neural Information Processing Systems*, 33:15859–15870, 2020.

Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., and Lee, S.-J. NOTE: Robust continual test-time adaptation against temporal correlation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=E9HNxrCFZPV.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. Improving robustness using generated data. In *Neural Information Processing Systems*, 2021.

Goyal, S., Sun, M., Raghunathan, A., and Kolter, Z. Test-time adaptation via conjugate pseudo-labels. *ArXiv*, abs/2207.09640, 2022.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *Arxiv*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hendrycks, D. and Dietterich, T. G. Benchmarking neural network robustness to common corruptions and perturbations. *ArXiv*, abs/1903.12261, 2019.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. Augmix: A simple data processing method to improve robustness and uncertainty. *ArXiv*, abs/1912.02781, 2020.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T. L., Parajuli, S., Guo, M., Song, D. X., Steinhardt, J., and Gilmer, J. The many faces of robustness: A critical analysis of out-of-distribution generalization. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8320–8329, 2021.

Hu, X., Uzunbas, M. G., Chen, S., Wang, R., Shah, A., Nevatia, R., and Lim, S.-N. Mixnorm: Test-time adaptation through online normalization estimation. *ArXiv*, abs/2110.11478, 2021.

Huang, H., Gu, X., Wang, H., Xiao, C., Liu, H., and Wang, Y. Extrapolative continuous-time bayesian neural network for fast training-free test-time adaptation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=wiHzQWwg3l.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Iwasawa, Y. and Matsuo, Y. Test-time classifier adjustment module for model-agnostic domain generalization. In *Neural Information Processing Systems*, 2021.

Jia, X., Zhang, Y., Wu, B., Ma, K., Wang, J., and Cao, X. Las-at: Adversarial training with learnable attack strategy. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13388–13398, 2022.

Kang, D., Sun, Y., Hendrycks, D., Brown, T. B., and Steinhardt, J. Testing robustness against unforeseen adversaries. *ArXiv*, abs/1908.08016, 2019.

Kantorovich, L. V. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pp. 199–201, 1942.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. *ArXiv*, abs/1703.04730, 2017.

Kojima, T., Matsuo, Y., and Iwasawa, Y. Robustifying vision transformer without retraining from scratch by test-time class-conditional feature alignment. *ArXiv*, abs/2206.13951, 2022.

Kundu, J. N., Venkat, N., RahulM., V., and Babu, R. V. Universal source-free domain adaptation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4543–4552, 2020.

Lee, D.-H. et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896, 2013.

Li, R., Jiao, Q., Cao, W., Wong, H.-S., and Wu, S. Model adaptation: Unsupervised domain adaptation without source data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9638–9647, 2020.

Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

Liang, J., Hu, D., and Feng, J. Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6028–6039. PMLR, 13–18 Jul 2020.

Liu, Y., Kothari, P., van Delft, B., Bellot-Gurlet, B., Mordan, T., and Alahi, A. Ttt++: When does self-supervised test-time training fail or thrive? In *Neural Information Processing Systems*, 2021.

Long, M., Cao, Z., Wang, J., and Jordan, M. I. Conditional adversarial domain adaptation. *Advances in neural information processing systems*, 31, 2018.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv: Learning*, 2016.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018.

Marchant, N. G., Rubinstein, B. I. P., and Alfeld, S. Hard to forget: Poisoning attacks on certified machine unlearning. In *AAAI Conference on Artificial Intelligence*, 2021.

Mehra, A., Kailkhura, B., Chen, P.-Y., and Hamm, J. Understanding the limits of unsupervised domain adaptation via data poisoning. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17347–17359. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/90cc440b1b8caa520c562ac4e4bbcb51-Paper.pdf.

Mirza, M. J., Micorek, J., Possegger, H., and Bischof, H. The norm must go on: Dynamic unsupervised domain adaptation by normalization. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14745–14755, 2021.

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

Nelson, B., Barreno, M., Chi, F. J., Joseph, A. D., Rubinstein, B. I. P., Saini, U., Sutton, C., Tygar, J. D., and Xia, K. Exploiting machine learning to subvert your spam filter. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., and Tan, M. Efficient test-time model adaptation without forgetting. In *The Internetional Conference on Machine Learning*, 2022.

Pang, T., Lin, M., Yang, X., Zhu, J., and Yan, S. Robustness and accuracy could be reconcilable by (proper) definition. In *International Conference on Machine Learning*, 2022.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. A. Fixing data augmentation to improve adversarial robustness. *ArXiv*, abs/2103.01946, 2021.

Rusak, E., Schneider, S., Pachitariu, G., Eck, L., Gehler, P. V., Bringmann, O., Brendel, W., and Bethge, M. If your data distribution shifts, use self-learning, 2022. URL https://openreview.net/forum?id=1oEvY1a67c1.

Saito, K., Watanabe, K., Ushiku, Y., and Harada, T. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3723–3732, 2018.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? *ArXiv*, abs/2007.08489, 2020.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. *ArXiv*, abs/2006.16971, 2020.

Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., and Mittal, P. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *International Conference on Learning Representations*, 2022.

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

Singh, S. and Shrivastava, A. Evalnorm: Estimating batch normalization statistics for evaluation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3632–3640, 2019.

Sinha, S., Gehler, P., Locatello, F., and Schiele, B. Test: Test-time self-training under distribution shift. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2759–2769, January 2023.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

Tramèr, F., Shokri, R., Joaquin, A. S., Le, H. M., Jagielski, M., Hong, S., and Carlini, N. Truth serum: Poisoning machine learning models to reveal their secrets. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

Vapnik, V. *Statistical learning theory*. John Wiley & Sons, 1998.

Vorobeychik, Y. and Kantarcioglu, M. *Adversarial Machine Learning*. Morgan Claypool, 2018.

Wang, D., Ju, A., Shelhamer, E., Wagner, D. A., and Darrell, T. Fighting gradients with gradients: Dynamic defenses against adversarial attacks. *ArXiv*, abs/2105.08714, 2021a.

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

Wang, H., Zhang, A., Zheng, S., Shi, X., Li, M., and Wang, Z. Removing batch normalization boosts adversarial training. In *ICML*, 2022a.

Wang, J.-K. and Wibisono, A. Towards understanding gd with hard and conjugate pseudo-labels for test-time adaptation. *ArXiv*, abs/2210.10019, 2022.

Wang, Q., Fink, O., Van Gool, L., and Dai, D. Continual test-time domain adaptation. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2022b.

Wang, X., Jin, Y., Long, M., Wang, J., and Jordan, M. I. Transferable normalization: Towards improving transferability of deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2019.

Wu, D., Xia, S., and Wang, Y. Adversarial weight perturbation helps robust generalization. *arXiv: Learning*, 2020a.

Wu, J. and He, J. Indirect invisible poisoning attacks on domain adaptation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, pp. 1852–1862, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467214. URL https://doi.org/10.1145/3447548.3467214.

Wu, Y.-H., Yuan, C.-H., and Wu, S.-H. Adversarial robustness via runtime masking and cleansing. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10399–10409. PMLR, 13–18 Jul 2020b. URL https://proceedings.mlr.press/v119/wu20f.html.

Xie, C., Zhang, Z., Wang, J., Zhou, Y., Ren, Z., and Yuille, A. L. Improving transferability of adversarial examples with input diversity. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2725–2734, 2018.

Xie, C., Wu, Y., van der Maaten, L., Yuille, A. L., and He, K. Feature denoising for improving adversarial robustness. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 501–509, 2019.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., and Le, Q. V. Adversarial examples improve image recognition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 816–825, 2020.

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

Zagoruyko, S. and Komodakis, N. Wide residual networks. *ArXiv*, abs/1605.07146, 2016.

Zhang, M., Levine, S., and Finn, C. Memo: Test time robustness via adaptation and augmentation. *ArXiv*, abs/2110.09506, 2021a.

Zhang, M., Marklund, H., Dhawan, N., Gupta, A., Levine, S., and Finn, C. Adaptive risk minimization: Learning to adapt to domain shift. In *NeurIPS*, 2021b.

Zhang, X., Gu, S. S., Matsuo, Y., and Iwasawa, Y. Domain prompt learning for efficiently adapting clip to unseen domains, 2021c. URL https://arxiv.org/abs/2111.12853.

Zhou, A. and Levine, S. Bayesian adaptation for covariate shift. *Advances in Neural Information Processing Systems*, 34:914–927, 2021.

A. Omitted Details in Background and Related Work

In this appendix, we cover additional details on batch normalization (Appendix A.1) and self-learning (Appendix A.2), along with TTA algorithms in Appendix A.3. Then, we discuss conventional machine learning vulnerabilities including adversarial examples and data poisoning in Appendix A.4. Furthermore, we include a comprehensive review of the existing literature in Appendix A.5.

A.1. Batch Normalization

In batch normalization, we calculate the BN statistics for each BN layer l by computing the mean μ and variance σ^2 of the pre-activations $\mathbf{z}_l$: $\mu \leftarrow \mathbb{E}[\mathbf{z}_l]$, $\sigma^2 \leftarrow \mathbb{E}[(\mathbf{z}_l - \mu)^2]$. The expectation $\mathbb{E}$ is computed channel-wise on the input. We then normalize the pre-activations by subtracting the mean and dividing by the standard deviation: $\bar{\mathbf{z}}_l = (\mathbf{z}_l - \mu)/\sigma$. Finally, we scale and shift the standardized pre-activations $\bar{\mathbf{z}}_l$ to $\mathbf{z}'_l = \gamma_l^T \bar{\mathbf{z}}_l + \beta_l$ by learnable affine parameters $\{\gamma_l, \beta_l\}$. In our experiments, the parameters updated during the TTA procedures are $\theta_{\mathcal{A}} = \{\gamma_l, \beta_l\}_{l=1}^{L}$, where L is the number of BN layers. The statistics of the source data are replaced with those of the test batch.

A.2. Objectives of Self-Learning

We provide detailed formulations of the TTA loss functions used in self-learning (SL) methods. Let $h(\cdot;\theta)$ be the hypothesis function parameterized by θ, which we consider to be the logit output (w.l.o.g.). The probability that sample $\mathbf{x}$ belongs to class j is denoted as $p(j|\mathbf{x}) = \sigma(h(\mathbf{x};\theta))$, where $\sigma(\cdot)$ is the softmax function. Here, we use θ to denote θ^{pre} for simplicity.

TENT (Wang et al., 2021b). This method minimizes the entropy of model predictions.

$$\mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t) := -\frac{1}{N_{Bt}} \sum_{i=1}^{N_{Bt}} \sum_j p(j|\mathbf{x}_i^t) \log p(j|\mathbf{x}_i^t) \tag{6}$$

Hard PL (Galstyan & Cohen, 2007; Lee et al., 2013). The most likely class predicted by the pre-adapted model is computed as the pseudo label for the unlabeled test data.

$$\begin{aligned} &\mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t, \widetilde{\mathbf{y}}(\theta)) := -\frac{1}{N_{Bt}} \sum_{i=1}^{N_{Bt}} \log p(\widetilde{y}_i(\theta)|\mathbf{x}_i^t), \\ &\text{where } \widetilde{y}_i(\theta) = \underset{j}{\text{argmax}} p(j|\mathbf{x}_i^t), \forall \mathbf{x}_i^t \in \mathbf{X}_B^t \end{aligned} \tag{7}$$

Soft PL (Galstyan & Cohen, 2007; Lee et al., 2013). Instead of using the predicted class, the softmax function is applied directly to the prediction to generate a pseudo label.

$$\begin{aligned} &\mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t, \widetilde{\mathbf{y}}(\theta)) := -\frac{1}{N_{Bt}} \sum_{i=1}^{N_{Bt}} \sum_j \widetilde{y}_i(\theta) \log p(j|\mathbf{x}_i^t), \\ &\text{where } \widetilde{y}_i(\theta) = p(j|\mathbf{x}_i^t), \forall \mathbf{x}_i^t \in \mathbf{X}_B^t \end{aligned} \tag{8}$$

Robust PL (Rusak et al., 2022). It has been shown that the cross-entropy loss is sensitive to label noise. To mitigate the side effect to the training stability and hyperparameter sensitivity, **Robust PL** replaces the cross-entropy (CE) loss of the **Hard PL** with a *Generalized Cross Entropy (GCE)*.

$$\begin{aligned} &\mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t, \widetilde{\mathbf{y}}(\theta)) := -\frac{1}{N_{Bt}} \sum_{i=1}^{N_{Bt}} q^{-1}(1 - p(\widetilde{y}_i(\theta)|\mathbf{x}_i^t)^q), \\ &\text{where } \widetilde{y}_i(\theta) = \underset{j}{\text{argmax}} p(j|\mathbf{x}), \forall \mathbf{x}_i^t \in \mathbf{X}_B^t \end{aligned} \tag{9}$$

where $q \in (0, 1]$ adjusts the shape of the loss function. When $q \to 1$, the GCE loss approaches the MAE loss, whereas when $q \to 0$, it reduces to the CE losses.

Conjugate PL (Goyal et al., 2022). We consider the source loss function, which can be expressed as $\mathcal{L}(h(\mathbf{x};\theta),y) = g(h(\mathbf{x};\theta)) - y^T h(\mathbf{x};\theta)$, where g is a function and y is a one-hot encoded class label. The TTA loss for self-learning with conjugate pseudo-labels can be written as follows:

$$\begin{aligned}&\mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t, \widetilde{\mathbf{y}}(\theta)) := -\frac{1}{N_{Bt}}\sum_{i=1}^{N_{Bt}} \mathcal{L}(h(\mathbf{x}_i^t;\theta)/T, \widetilde{y}_i(\theta)),\\ &\text{where } \widetilde{y}_i(\theta) = \nabla g(h(\mathbf{x}_i^t)/T;\theta), \forall \mathbf{x}_i^t \in \mathbf{X}_B^t\end{aligned} \tag{10}$$

The temperature T is used to scale the predictor.

A.3. TTA algorithms in details

In this subsection, we present the detailed algorithm of test-time adaptation (Algorithm 2). In our setting, the model is adapted online when a batch of test data comes and then makes the prediction immediately.

Algorithm 2 Test-Time Adaptation

Input: Source model parameters θ^s, number of steps N, TTA update rate: η
Initialization: $\theta = \theta^s$
for step $= 1, 2, \ldots, N$ **do**
  Obtain a new test batch $\mathbf{X}_B^t$ from the test domain.
  $\theta_{\mathcal{B}} \leftarrow \{\mu(\mathbf{X}_B^t), \sigma^2(\mathbf{X}_B^t)\}$
  $\theta_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} - \eta \cdot \partial \mathcal{L}_{\text{TTA}}(\mathbf{X}_B^t)/\partial \theta_{\mathcal{A}}$
  $\theta \leftarrow \theta_{\mathcal{A}} \cup \theta_{\mathcal{B}} \cup \theta_{\mathcal{F}}$
  Make the prediction $\hat{\mathbf{y}}_B^t = f(\mathbf{X}_B^t;\theta)$
end for

A.4. Conventional Machine Learning Vulnerabilities

The following subsection provides additional background on ML vulnerabilities and their relevance to DIA. We then emphasize the distinctiveness of the proposed attack, which differentiates it from other known adversarial examples and data poisoning attacks.

Adversarial Examples. The vast majority of work studying vulnerabilities of deep neural networks concentrates on finding the imperceptible *adversarial examples*. Those examples have been successfully used to fool the model at test time (Goodfellow et al., 2015; Carlini & Wagner, 2017; Vorobeychik & Kantarcioglu, 2018). Commonly, generating an adversarial example $\hat{\mathbf{x}}^t = \mathbf{x}^t + \hat{\boldsymbol{\delta}}$ can be formulated as follows:

$$\hat{\boldsymbol{\delta}} = \arg\max_{\boldsymbol{\delta}} \mathcal{L}(f(\mathbf{x}^t + \boldsymbol{\delta};\theta^s), y^t) \quad s.t. \left\|\boldsymbol{\delta}\right\|_p \leq \varepsilon, \tag{11}$$

where $\mathcal{L}$ is the loss function (e.g., cross-entropy loss), $\mathbf{x}^t, y^t$ denote one test data, $\left\|\boldsymbol{\delta}\right\|_p \leq \varepsilon$ is the ℓ_p constraint of the perturbation $\boldsymbol{\delta}$. The objective is to optimize $\boldsymbol{\delta}$ to maximize the prediction loss $\mathcal{L}$. For ℓ_∞ constraint, the problem can be solved by Projected Gradient Descent (Madry et al., 2018) as

$$\boldsymbol{\delta} \leftarrow \Pi_\varepsilon(\boldsymbol{\delta} + \alpha \operatorname{sign}(\nabla_{\boldsymbol{\delta}} L(f(\mathbf{x}^t + \boldsymbol{\delta};\theta^s), y^t))). \tag{12}$$

Here, Π_ε denotes projecting the updated perturbation back to the constraint, and α is the attack step size. By iteratively updating through Eq. (12), the attacker will likely cause the model to mispredict. Our single-level variant of the DIA attack employs an algorithm similar to the projected gradient descent.

One characteristic of *adversarial examples* in ML pipeline is that the perturbations have to be made on the targeted data. Therefore, as long as the user keeps the test data securely (not distorted), the machine learning model can always make benign predictions. Our attack, targeted at TTA, differs from adversarial examples, where our malicious samples can be

inserted into a test batch and attack the benign and unperturbed data. Therefore, there exists an increasing security risk if deploying TTA.

Data Poisoning Attacks. In terms of attacking benign samples without directly modifying the data, another pernicious attack variant, *data poisoning*, can accomplish this goal. Specifically, an adversary injects poisoned samples into a training set and aims to cause the trained model to mispredict the benign sample. There are two main objectives for a poisoning attack: targeted poisoning and indiscriminate. For *targeted poisoning attacks* (Koh & Liang, 2017; Carlini & Terzis, 2022; Carlini, 2021; Geiping et al., 2020), the attacker's objective is to attack one particular sample with a pre-selected targeted label, which can be written as follows:

$$\min_{\boldsymbol{\delta}} \mathcal{L}\left(f\left(\mathbf{x}_{tgt}^{t}, \theta^{*}(\boldsymbol{\delta})\right), \dot{y}_{tgt}\right) \quad s.t.\ \theta^{*}(\boldsymbol{\delta}) = \arg\min_{\theta} \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{L}\left(f\left(\mathbf{x}_{i}^{s} + \boldsymbol{\delta}_i, \theta\right), y_i^s\right) \tag{13}$$

where $\mathbf{x}_{tgt}^{t}$ is the targeted samples and $\dot{y}_{tgt}$ is the correpsonding incorrect targeted labels. $\mathbf{x}_i^s$ is the training data and y_i^s is the corresponding ground truth. We use the $\boldsymbol{\delta}_i$ to denote the perturbation on training data $\mathbf{x}_i^s$, and since there are N_m poisoning samples, some $\boldsymbol{\delta}_i$ stays zero to represent clean samples.

For *indiscriminate attacks* (Nelson et al., 2008; Biggio et al., 2012), the adversary seeks to reduce the accuracy of all test data, which can be formulated as:

$$\min_{\boldsymbol{\delta}} -\frac{1}{N_t} \sum_{j=1}^{N_t} \mathcal{L}\left(f\left(\mathbf{x}_{j}^{t}, \theta^{*}(\boldsymbol{\delta})\right), y_j^t\right) \quad s.t.\ \theta^{*}(\boldsymbol{\delta}) \in \arg\min_{\theta} \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{L}\left(f\left(\mathbf{x}_{i}^{t} + \boldsymbol{\delta}_i, \theta\right), y_i^t\right) \tag{14}$$

where $\mathbf{x}_j^t$ is the test data and y_j^t is the ground truth of it. These objectives also guide the design of our adversary's goals.

However, the key characteristic of data poisoning for the machine learning model is the assumption of access to training data for adversaries. Therefore, if the model developer obtains the training data from a reliable source and keeps the database secure, then there is no chance for any poisoning attacks to be effective. Our attack for TTA differs from data poisoning, in which DIA only requires test data access. This makes our attack easier to access, as data in the wild environment (test data) are less likely to be monitored.

A.5. Extended Related Work

We present more related works of our paper in this subsection.

Unsupervised Domain Adaptation. Unsupervised domain adaptation (UDA) deals with the problem of adapting a model trained on a source domain to perform well on a target domain, using both labeled source data and unlabeled target data. One common approach (Ganin et al., 2016; Long et al., 2018) is to use a neural network with shared weights for both the source and target domains and introduce an additional loss term to encourage the network to learn domain-invariant features. Other approaches include explicitly (Saito et al., 2018; Damodaran et al., 2018) or implicitly (Li et al., 2016; Wang et al., 2019) aligning the distributions of the source and target domains. One category related to our settings is source-free domain adaptation (Liang et al., 2020; Kundu et al., 2020; Li et al., 2020), where they assume a source model and the entire test set. For instance, SHOT (Liang et al., 2020) uses information maximization and pseudo-labels to align the source and target domain during the inference stage.

Test-time Adaptation. TTA obtains the model trained on the source domain and performs adaptation on the target domain. Some methods (Sun et al., 2020; Liu et al., 2021; Gandelsman et al., 2022) modify the training objective by adding a self-supervised proxy task to facilitate test-time training. However, in many cases, access to the training process is unavailable. Therefore some methods for test-time adaptation only revise the Batch Norm (BN) statistics (Singh & Shrivastava, 2019; Nado et al., 2020; Schneider et al., 2020; You et al., 2021; Hu et al., 2021). Later, TENT (Wang et al., 2021b), BACS (Zhou & Levine, 2021), and MEMO (Zhang et al., 2021a) are proposed to improve the performance by minimizing entropy at test time. Other approaches (Galstyan & Cohen, 2007; Lee et al., 2013; Rusak et al., 2022; Goyal et al., 2022; Wang & Wibisono, 2022) use pseudo-labels generated by the source (or other) models to self-train an adapted model. As an active research area, recent efforts have been made to further improve TTA methods in various aspects. For example, Niu et al. (2022) seeks to solve the forgetting problem of TTA. Wang et al. (2022b); Gong et al. (2022); Huang et al. (2022) propose methods to

address the continual domain shifts. Iwasawa & Matsuo (2021) improves the pseudo-labels by using the pseudo-prototype representations. Zhang et al. (2021c); Kojima et al. (2022); Gao et al. (2022b) leverage recent vision transformer model architecture to improve TTA.

Most TTA methods assume a batch of test samples is available, while some papers consider the test samples coming individually (Zhang et al., 2021a; Gao et al., 2022a; Bartler et al., 2022; Mirza et al., 2021; Döbler et al., 2022). For instance, MEMO (Zhang et al., 2021b) leverages various augmentations on single test input and then adapts the model parameters. Gao et al. (2022a) propose using diffusion to convert the out-of-distribution samples back to the source domain. Such a "single-sample" adaption paradigm makes an independent prediction on each data, avoiding the risk (i.e., DIA) of TTA. However, batch-wise test samples provide more information about the distribution, usually achieving better performance.

Adversarial Examples and Defenses. A host of works have explored the imperceptible adversarial examples (Goodfellow et al., 2015; Carlini & Wagner, 2017; Vorobeychik & Kantarcioglu, 2018) which fool the model at test time and raise security issues. In response, adversarial training (Madry et al., 2017) (training with adversarial samples) has been proposed as an effective technique for defending against adversarial examples. Later, there existed a host of enhanced methods on robustness (Wu et al., 2020a; Sehwag et al., 2022; Dai et al., 2022; Gowal et al., 2021), time efficiency (Shafahi et al., 2019; Wong et al., 2019), and utility-robustness tradeoff (Pang et al., 2022; Wang et al., 2022a).

Data Poisoning Attacks and Defenses. Data poisoning attacks refer to injecting poisoned samples into a training set and causing the model to predict incorrectly. It has been applied to different machine learning algorithms, including the Bayesian-based method (Nelson et al., 2008), support vector machines (Biggio et al., 2012), as well as neural networks (Koh & Liang, 2017). Later, Carlini (2021) tries to poison the unlabeled training set of semi-supervised learning and Carlini & Terzis (2022) target at contrastive learning with an extremely limited number of poisoning samples. Recently, researchers have utilized poisoning attacks to achieve other goals (e.g., enhancing membership inference attacks (Tramèr et al., 2022) and breaking machine unlearning (Marchant et al., 2021; Di et al., 2022)). For defense, Geiping et al. (2021) has shown that adversarial training can also effectively defend against data poisoning attacks.

Adversarial Risk in Domain Adaptation. Only a few papers discuss adversarial risks in the context of domain adaptation settings. One example is (Mehra et al., 2021), which proposes several methods for adding the poisoned data to the source domain data exploiting both the source and target domain information. Another example is (Wu & He, 2021), which introduces *I2Attack*, an indirect, invisible poisoning attack that only manipulates the source examples but can cause the domain adaptation algorithm to make incorrect predictions on the targeted test examples. However, both approaches assume that the attacker has access to the source data and cannot be applied to the source-free (TTA) setting.

B. Theorical Analysis of Distribution Invading Attacks

In this appendix, we provide technical details of computing the DIA gradient described in Section 4. Then, we analyze why smoothing via training-time BN can mitigate our attacks. For simplicity, we consider the case for a single layer, and the gradient computation can be generalized to the case of a multi-layer through backpropagation.

B.1. Understanding the Vulnerabilities of Re-estimating BN Statistics

Recall the setting where $\mathbf{x}_{tgt}^t \in \mathbb{R}^d$, $\mathbf{X}_B^t = (x_{i,j})_{i=1\ldots n, j=1\ldots d} \in \mathbb{R}^{n\times d}$, $\vec{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. The output of a linear layer with batch normalization[6] on a single dimension can be written as

$$f(\mathbf{x}_{tgt}^t) = \frac{\mathbf{x}_{tgt}^t - \mu(\mathbf{X}_B^t)}{\sqrt{\sigma^2(\mathbf{X}_B^t)}}\vec{w} + b \tag{15}$$

where μ and $\sigma^2 : \mathbb{R}^{n\times d} \to \mathbb{R}^d$ denote the coordinate-wise mean and variance. In addition, the square root $\sqrt{\cdot}$ is also applied coordinate-wise. Suppose we can perturb x_i in $\mathbf{X}_B^t$, which corresponds to a malicious data point in $\mathbf{X}_{mal}^t$. In order to find the optimal perturbation direction that causes the largest deviation in the prediction of $f(\mathbf{x}_{tgt}^t)$, we compute the following derivative:

$$\begin{aligned}
\frac{\partial f(\mathbf{x}_{tgt}^t)}{\partial x_{i,j^*}} &= \frac{(-\frac{1}{n})(\sigma(\mathbf{X}_B^t))_j - [(\mathbf{x}_{tgt}^t)_j - \mu(\mathbf{X}_B^t)_j]\frac{\partial(\sigma(\mathbf{X}_B^t))_j}{\partial x_{i,j}}}{(\sigma(\mathbf{X}_B^t))_j^2} w_j \\
&= \frac{(-\frac{1}{n})\left\{(\sigma(\mathbf{X}_B^t))_j - \left[(\mathbf{x}_{tgt}^t)_j - \mu(\mathbf{X}_B^t)_j\right](\sigma(\mathbf{X}_B^t))_j^{-1}(x_{i,j^*} - \mu(\mathbf{X}_B^t)_j)\right\}}{(\sigma(\mathbf{X}_B^t))_j^2} w_j \\
\text{where}\quad (\mu(\mathbf{X}_B^t))_j &= \frac{1}{n}\sum_{i=1}^n x_{i,j} \\
(\sigma(\mathbf{X}_B^t))_j &= \sqrt{\frac{1}{n}\sum_{i=1}^n x_{i,j}^2 - \left(\frac{1}{n}\sum_{i=1}^n x_{i,j}\right)^2}
\end{aligned}$$

Then, we can leverage the above formula to search for the optimal malicious data.

B.2. Analysis of Smoothing via Training-time BN Statistics

We robustly estimate the final BN statistics by $\bar{\mu} = \tau\vec{\mu}_s + (1-\tau)\vec{\mu}_t, \bar{\sigma}^2 = \tau\vec{\sigma}_s^2 + (1-\tau)\vec{\sigma}_t^2$, where $(\vec{\mu}_s, \vec{\sigma}_s^2)$ are training-time BN and $(\vec{\mu}_t = \mu(\mathbf{X}_B^t), \vec{\sigma}_t^2 = \sigma^2(\mathbf{X}_B^t))$ are test-time BN (shown in Section 6.2). Therefore, the output of a linear layer with smoothed batch normalization on a single dimension can be re-written as

$$f(\mathbf{x}_{tgt}^t) = \frac{\mathbf{x}_{tgt}^t - ((1-\tau)\mu(\mathbf{X}_B^t) + \tau\vec{\mu}_s)}{\sqrt{(1-\tau)\sigma^2(\mathbf{X}_B^t) + \tau\vec{\sigma}_s^2}}\vec{w} + b \tag{16}$$

The new gradient can be computed by

[6]For the purpose of this analysis, we will set the scale parameter to 1 and the shift parameter to 0. This assumption does not limit the generalizability of our results.

$$\frac{\partial f(\mathbf{x}_{tgt}^t)}{\partial x_{i,j^*}} = \frac{(-\frac{1-\tau}{n})\tilde{\sigma}_j - [(\mathbf{x}_{tgt}^t)_j - (1-\tau)\mu(\mathbf{X}_B^t)_j - \tau(\vec{\mu}_s)_j]\frac{1}{2}(\tilde{\sigma}_j)^{-1}(1-\tau)\frac{\partial(\sigma(\mathbf{X}_B^t))_j}{\partial x_{i,j}}}{(1-\tau)(\sigma(\mathbf{X}_B^t))_j^2 + \tau(\vec{\sigma}_s^2)_j} w_j$$

$$= \frac{(-\frac{1-\tau}{n})\left\{(\sigma(\mathbf{X}_B^t))_j^2 - \left[(\mathbf{x}_{tgt}^t)_j - (1-\tau)(\mu(\mathbf{X}_B^t))_j - \tau(\vec{\mu}_s)_j\right](\tilde{\sigma}_j)^{-1}(x_{i,j^*} - (\mu(\mathbf{X}_B^t))_j)\right\}}{(1-\tau)(\sigma(\mathbf{X}_B^t))_j^2 + \tau(\vec{\sigma}_s^2)_j} w_j$$

where $(\mu(\mathbf{X}_B^t))_j = \frac{1}{n}\sum_{i=1}^n x_{i,j}$,

$$(\sigma(\mathbf{X}_B^t))_j = \sqrt{\frac{1}{n}\sum_{i=1}^n x_{i,j}^2 - (\frac{1}{n}\sum_{i=1}^n x_{i,j})^2}$$

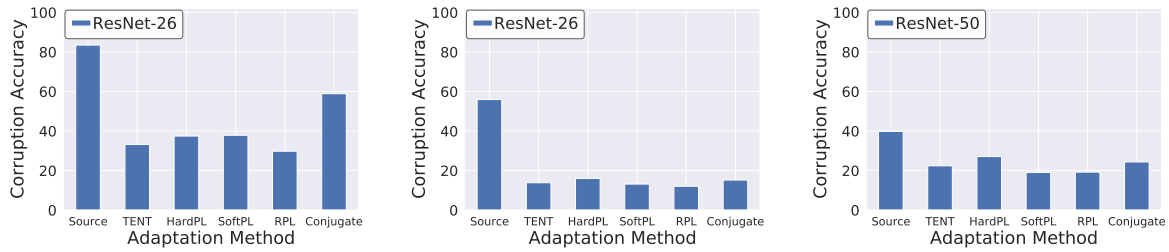
$$\tilde{\sigma}_j := \sqrt{(1-\tau)(\sigma(\mathbf{X}_B^t))_j^2 + \tau(\vec{\sigma}_s^2)_j}.$$

Here, whenever $\mu(\mathbf{X}_B^t) \approx \vec{\mu}_s$, and $\sigma^2(\mathbf{X}_B^t) \approx \vec{\sigma}_s^2$, the norm of graident $||\frac{\partial f(\mathbf{x}_{tgt}^t)}{\partial x_{i,j^*}}||$ will decrease as τ increases. Specifically, $||\frac{\partial f(\mathbf{x}_{tgt}^t)}{\partial x_{i,j^*}}|| = 0$ when $\tau = 1$, which means the BN statistics is only based on the training-time BN and do not involve any adversarial risks.

C. Preliminary Results

C.1. Sanity Check of Utilizing Training-time BN Statistics for TTA

In this experiment, we evaluate the performance of TTA using training-time BN statistics on CIFAR-10-C, CIFAR-100-C, and ImageNet-C with the ResNet architecture (He et al., 2016). Note that we did not include the result of the **TeBN** method, as it is identical to the **Source** method (directly using the source model without any TTA method). For other TTA methods, only affine transformation parameters (i.e., scale γ_l and shift β_l) within the BN layer are updated. All other hyperparameters stay the same with experiments in Goyal et al. (2022), including batch size 200 and TTA learning rate $\eta = 0.001$.



(a) CIFAR-10-C (b) CIFAR-100-C (c) ImageNet-C

Figure 9. Test-time adaptations exhibit an obvious degradation on corruption datasets when using training-time BN statistics. [Severity Level: 3]

Adopting training-time BN completely ruins performance gain for TTA (Figure 9). We observe that all TTA methods exhibit a significant degradation (from ∼15% to ∼50%) across benchmarks. It can be inferred that the utilization of test-time batch normalization statistics is paramount to all TTA methods implemented.

C.2. Preliminary Results of Bilevel Optimization

We study the effectiveness of utilizing bilevel optimization to find the malicious data $\widehat{\mathbf{X}}_{mal}^{t}$.

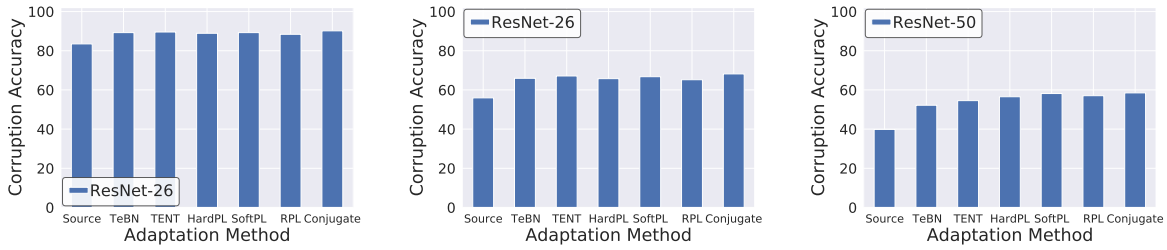
Table 6. Attack success rate of Distribution Invading Attack with and without bilevel optimization. (We omit **TeBN** as it is identical for both methods)

Dataset	N_m	Bilevel	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
CIFAR-10-C (ResNet-26)	10 (5%)	✗	23.20	25.33	23.20	24.80	23.60
	10 (5%)	✓	22.93	23.73	22.80	24.00	23.73
	20 (10%)	✗	45.73	48.13	47.47	49.47	45.73
	20 (10%)	✓	44.40	47.33	46.40	47.60	45.20
	40 (10%)	✗	83.87	84.27	82.93	86.93	85.47
	40 (10%)	✓	82.80	84.53	82.40	84.53	84.67
CIFAR-100-C (ResNet-26)	10 (5%)	✗	26.40	31.20	27.60	32.13	26.13
	10 (5%)	✓	26.93	31.87	28.80	32.93	26.40
	20 (10%)	✗	72.80	87.33	78.53	82.93	71.60
	20 (10%)	✓	72.27	85.87	78.13	85.47	72.13
	40 (10%)	✗	100.00	99.87	100.00	99.87	100.00
	40 (10%)	✓	100.00	99.73	100.00	100.00	100.00
ImageNet-C (ResNet-50)	5 (2.5%)	✗	75.73	69.87	62.67	66.40	57.87
	5 (2.5%)	✓	74.93	70.13	62.13	67.47	59.43
	10 (5%)	✗	98.67	96.53	94.13	96.00	92.80
	10 (5%)	✓	98.40	97.07	94.40	96.27	93.43
	20 (10%)	✗	100.00	100.00	100.00	100.00	100.00
	20 (10%)	✓	100.00	100.00	100.00	99.73	100.00

Our proposed single-level solution yields comparable effectiveness to the Bi-level Optimization approach (Table 6). We evaluate the DIA method with and without the bilevel optimization (inner loop) step, revealing that, in most cases, the difference between them is generally less than 1%. In addition, we observe that using bilevel optimization results in a substantial increase (by a factor of $\sim$10) in computational time.

C.3. Effectiveness of Test-time Adaptations on Corruption Datasets

We demonstrate the effectiveness of applying TTA methods to boost corruption accuracy. We use **Source** to denote the performance of the source model without TTA.



(a) CIFAR-10-C (b) CIFAR-100-C (c) ImageNet-C

Figure 10. Test-time adaptations consistently improve the accuracy on corruption datasets with severity level 3.

All TTA methods significantly boost the accuracy on distribution shift benchmarks (Figure 10). We observe the absolute performance gains of TTA on corrupted inputs are $>$**5%** for CIFAR-10-C, $>$**10%** for CIFAR-100-C, and $>$**12%** for ImageNet-C. These promising results incentivize the use of TTA methods in various applications when test data undergoes a shift in distribution.

D. Additional Experiment Details and Results of Distribution Invading Attacks

In this appendix, we show some supplementary experimental details and results in Section 5. We begin by outlining our experimental setup in Appendix D.1. This is followed by the experiments that study the effectiveness of DIA by varying the number of malicious samples in Appendix D.2 and using various models and data augmentations in Appendix D.3. Then, we delve into different types of attack objectives and constraints, including (1) indiscriminate attack (Appendix D.4), (2) stealthy targeted attack (Appendix D.5), and (3) distribution invading attack via simulated corruptions (Appendix D.6). In addition, we conduct ablation studies including different assumptions of the attacker's knowledge (Appendix D.7) and various batch sizes (Appendix D.8). Besides, we examine various TTA design choices, such as larger learning rates (Appendix D.10), and increased optimization steps (Appendix D.11). We also consider the DIA under various corruption choices by adjusting the severities of corruption in Appendix D.9 and present the detailed results of all types of corruption in Appendix D.13. Other new baseline methods are applied in Appendix D.12.

D.1. Addtional details of Experiment setup

Dataset. Our attacks are evaluated on CIFAR-10 to CIFAR-10-C, CIFAR-100 to CIFAR-100-C, and ImageNet to ImageNet-C (Hendrycks & Dietterich, 2019), which contain 15 types of corruptions on test data. We select all corruption types and set the severity of the corruption as 3 for most experiments.[7] Therefore, our CIFAR-10-C and CIFAR-100-C evaluation sets contain 10,000 $\times$ 15 = 150,000 images with 32×32 resolution from 10 and 100 classes, respectively. For ImageNet-C, there are 5,000 $\times$ 15 = 75,000 high-resolution (224×224) images from 1000 labels to evaluate our attacks.

Source Model Details. We follow the common practice on distribution shift benchmarks (Hendrycks & Dietterich, 2019), and train our models on the CIFAR-10, CIFAR-100, and ImageNet. For training models on the CIFAR dataset, we use the SGD optimizer with a 0.1 learning rate, 0.9 momentum, and 0.0005 weight decay. We train the model with a batch size of 256 for 200 epochs. We also adjust the learning rate using a cosine annealing schedule (Loshchilov & Hutter, 2016). This shares the same configurations with Goyal et al. (2022). As we mentioned previously, the architectures include ResNet with 26 layers (He et al., 2016), VGG 19 with layers (Simonyan & Zisserman, 2015), Wide ResNet with 28 layers (Zagoruyko & Komodakis, 2016). For the ImageNet benchmark, we directly utilize the models downloaded from RobustBench (Croce et al., 2020) (https://robustbench.github.io/), including standard trained ResNet-50 (He et al., 2016), AugMix (Hendrycks et al., 2020), DeepAugment (Hendrycks et al., 2021), robust models from Salman et al. (2020), and Engstrom et al. (2019). We want to emphasize that TTA does not necessarily need to train a model from scratch, and downloading from outsourcing is acceptable and sometimes desirable.

Test-time Adaptation Methods. Six test-time adaptation methods are selected, including **TeBN** (Schneider et al., 2020), **TENT** (Wang et al., 2021b), **Hard PL** (Lee et al., 2013; Galstyan & Cohen, 2007), **Soft PL** (Lee et al., 2013; Galstyan & Cohen, 2007), **Robust PL** (Rusak et al., 2022), and **Conjugate PL** (Goyal et al., 2022), where they all obtain obvious performance gains when data distribution shifts. We use the same default hyperparameters with Wang et al. (2021b) and Goyal et al. (2022), where the code is available at github.com/DequanWang/tent and github.com/locuslab/tta-conjugate. Besides setting the batch size to **200**, we use Adam for the TTA optimizer, $\eta = 0.001$ for the TTA learning rate, and 1 for temperature. TTA is done in 1 step for each test batch.

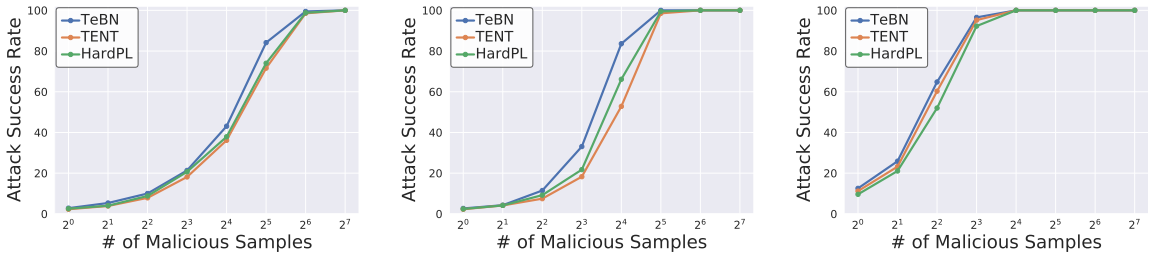
Attack Setting. Each test batch is an individual attacking trial, which contains 200 samples. Therefore, there are 150,000/200 = 750 trials for CIFAR-C and 75,000/200 = 375 trials for ImageNet-C. All of our experimental results are averaged across all trials. For targeted attacks, we use **attack success rate (ASR)** as the evaluation metrics for attack effectiveness, which means the ratio of DIA can flip the targeted sample to the pre-select label. For indiscriminate attacks, we use corruption **corruption error rate** (i.e., the error rate on the benign corrupted data) to measure the effectiveness. For stealthy targeted attacks, we again use **attack success rate (ASR)** as the attacking effectiveness metrics. In addition, since we want to maintain the corruption accuracy, we leverage the **corruption accuracy degradation** (i.e., the accuracy drop of benign corrupted data compared to "no attack") as the metric.

For other hyperparameters, we set the attacking steps $N = 500$ and attacking optimization rate $\alpha = 1/255$. In addition, the attack effectiveness can be further improved if applying more iterations or multiple random initializations like Madry et al. (2018). We leave more enhancing DIA techniques as future directions.

[7] the severities of the corruption range from 1 to 5, and 3 can be considered as medium corruption degree.

D.2. More Experiments of DIA across Number of Malicious Samples

Figure 11 depicts the effectiveness of the DIA success rate in relation to the number of malicious samples. We selected **TeBN**, **TENT**, and **Hard PL** as demonstrated TTA methods and observed them perform similarly. Our conclusion "DIA obtains near-100% ASR, using 64 malicious samples for CIFAR-10-C, 32 for CIFAR-100-C, and 16 for ImageNet-C" still holds.



(a) CIFAR-10-C (b) CIFAR-100-C (c) ImageNet-C

Figure 11. Success rate of our proposed attack across numbers of malicious samples from 1 to 128 (0.5% to 64%). [TTA: **TeBN**, **TENT**, and **Hard PL**] (Extended version of Figure 2)

D.3. Effectiveness of DIA across Various Model Architectures and Data Augmentations

Then we conduct more comprehensive experiments on DIA across model architectures, data augmentations, and the number of malicious samples. In Table 7 (full version of Table 2), we observe that the attack success rate of the VGG (Simonyan & Zisserman, 2015) architecture is less affected by the number of malicious samples compared to the Wide ResNet (Simonyan & Zisserman, 2015) architecture. For example, the ASR of VGG improves from $\sim$42% to $\sim$79% while Wide ResNet improves from $\sim$16% to $\sim$94% if N_m increases from 10 to 40 for the **TeBN** method on CIFAR-10-C. Therefore, our previous hypothesis should be modified to "**with a sufficient number of malicious data**, more BN layers expose more vulnerabilities." For ImageNet-C, we are still observing the mild mitigation effect from strong data augmentations.

D.4. Effectiveness of Indiscriminate Attack

We further present detailed results of indiscriminate attacks with more TTA methods and options for the number of malicious samples. Since the benign **corruption error rates** are different for different numbers of malicious data, we include the corruption error rate improvement in the bracket with red color.

Table 8. Average corruption error rate of TTA when deploying indiscriminate attack. The red number inside the bracket is the corruption error rate improvement. (Extended version of Table 3)

Dataset	N_m	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
CIFAR-10-C (ResNet26)	10 (5%)	15.66 (+4.96)	15.16 (+4.74)	15.86 (+4.84)	15.56 (+4.86)	16.60 (+5.04)	14.32 (+4.51)
	20 (10%)	20.09 (+9.37)	19.43 (+9.00)	20.14 (+9.10)	19.77 (+9.06)	21.04 (+9.46)	18.51 (+8.70)
	40 (20%)	28.02 (+17.29)	27.01 (+16.54)	27.91 (+16.86)	27.47 (+16.69)	28.81 (+17.20)	26.09 (+16.26)
CIFAR-100-C (ResNet26)	10 (5%)	43.84 (+7.22)	42.90 (+7.32)	44.65 (+7.92)	43.45 (+7.56)	45.34 (+8.02)	42.01 (+7.53)
	20 (10%)	51.99 (+13.44)	50.33 (+12.60)	52.00 (+13.33)	50.80 (+12.82)	52.52 (+13.21)	49.39 (+12.71)
	40 (20%)	58.41 (+23.89)	54.44 (+21.13)	56.59 (+21.93)	54.93 (+21.22)	56.97 (+21.72)	53.33 (+21.06)
ImageNet-C (ResNet50)	5 (2.5%)	57.42 (+9.45)	54.53 (+8.90)	52.32 (+8.70)	50.53 (+8.50)	51.52 (+8.43)	50.18 (+8.52)
	10 (5%)	67.18 (+19.21)	62.39 (+16.76)	58.70 (+15.09)	55.90 (+13.88)	57.08 (+13.97)	55.60 (+13.94)
	20 (10%)	79.03 (+31.24)	75.01 (+29.57)	70.26 (+26.84)	65.67 (+23.81)	67.53 (+24.60)	65.30 (+23.80)

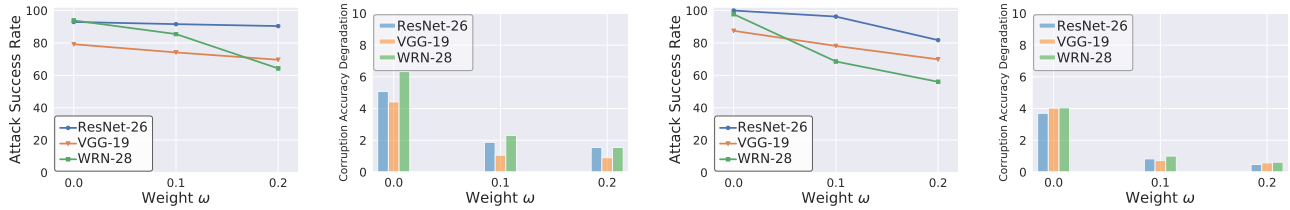
The indiscriminate DIA causes a large error rate increase on benign samples (Table 8). We observe that the ratio between the fraction of malicious data and error rate improvement is about 0.9$\times$ for CIFAR-10-C and 1.2$\times$ for CIFAR-100-C. For example, the error rate increases $\sim$12% with 20 (10%) malicious data for CIFAR-100-C. Significantly, 20 (10%) malicious data cause the error rate improves $\sim$25% for ImageNet-C. In addition, compared to other TTA methods, **TeBN** is still the most vulnerable method against DIA.

Table 7. Effectiveness of distribution invading attack across various model architectures, data augmentation, and the number of malicious data. (Full version of Table 2)

Dataset	N_m	Architectures	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
CIFAR-10-C	10 (5%)	ResNet-26	25.87	23.20	25.33	23.20	24.80	23.60
		VGG-19	41.73	30.27	30.13	28.00	33.07	29.07
		WRN-28	16.27	14.40	14.67	14.53	15.73	13.87
	20 (10%)	ResNet-26	55.47	45.73	48.13	47.47	49.47	45.73
		VGG-19	60.13	44.80	46.67	44.13	46.67	44.67
		WRN-28	46.00	41.47	43.33	41.60	44.00	40.53
	40 (20%)	ResNet-26	92.80	83.87	84.27	82.93	86.93	85.47
		VGG-19	79.07	65.33	67.47	64.13	67.47	64.13
		WRN-28	93.73	89.60	90.80	90.13	91.20	89.33
CIFAR-100-C	10 (5%)	ResNet-26	46.80	26.40	31.20	27.60	32.13	26.13
		VGG-19	42.13	32.00	41.33	33.60	33.87	37.33
		WRN-28	29.60	14.67	16.53	15.47	18.13	14.53
	20 (10%)	ResNet-26	93.73	72.80	87.33	78.53	82.93	71.60
		VGG-19	71.60	57.33	71.60	61.87	63.60	66.13
		WRN-28	64.80	38.67	44.13	40.93	46.40	39.87
	40 (20%)	ResNet-26	100.00	100.00	99.87	100.00	99.87	100.00
		VGG-19	88.00	82.93	86.53	85.73	87.33	85.47
		WRN-28	97.47	83.60	87.07	86.67	90.13	79.73
Dataset	N_m	**Augmentations**	**TeBN(%)**	**TENT(%)**	**Hard PL(%)**	**Soft PL(%)**	**Robust PL(%)**	**Conjugate PL(%)**
ImageNet-C	5 (2.5%)	Standard	80.80	75.73	69.87	62.67	66.40	57.87
		AugMix	72.53	65.60	59.47	53.60	56.27	49.07
		DeepAugment	67.20	63.73	58.67	53.87	57.33	53.33
	10 (5%)	Standard	99.47	98.67	96.53	94.13	96.00	92.80
		AugMix	98.40	96.00	93.60	88.27	92.00	87.20
		DeepAugment	96.00	94.67	91.20	87.47	89.60	86.67
	20 (10%)	Standard	100.00	100.00	100.00	100.00	100.00	100.00
		AugMix	100.00	100.00	100.00	100.00	100.00	100.00
		DeepAugment	100.00	100.00	100.00	100.00	100.00	100.00

D.5. Effectiveness of Stealthy Targeted Attack

Next, we present more results of the stealthy targeted attack on the CIFAR-C dataset in Figure 12. We use 40 malicious data and **TeBN** method, where $\omega = \{0, 0.1, 0.2\}$. Specifically, we select three architectures, including ResNet-26, VGG-19, and WRN-28. Since different architectures result in various corruption accuracy, we use the **corruption accuracy degradation** to measure the stealthiness, where the best attack should have 0% degradation. We observe that we can obtain $< 2\%$ degradation by sacrificing the attack success rates about 10% when $\omega = 0.1$. In addition, we find the attack success rates of Wide ResNet-28 drop much more than other model architectures.



(a) CIFAR-10-C (b) CIFAR-100-C

Figure 12. Adjusting the weight ω achieves a high attack success rate (line) and a minimal benign corruption accuracy degradation (bar). [N_m=40, TTA method: Norm] (Extended version of Figure 3)

D.6. Distribution Invading Attack via Simulated Corruptions

This subsection presents the details of generating simulated corruption and then demonstrates its effectiveness.

Constructing Simulated Corruptions. We directly utilize the methods of Kang et al. (2019) to construct adversarially snow and fog effects, which are available at github.com/ddkang/advex-uar. Specifically, we use many tiny occluded image regions representing snowflakes at randomly chosen locations. Then, the intensity and directions of those "snow pieces" are adversarially optimized. Furthermore, the adversarial fog is generated by adversarially optimizing the diamond-square algorithm (Fournier et al., 1982), a technique commonly used to create a random, stochastic fog effect. We present some snow and fog attack examples in Figure 27.

Simulated corruptions can be used as an input-stealthy DIA approach (Table 9). For evaluation, we use snow to attack clean images and insert them into the validation set of snow corruption. Similarly, we use a fog attack for the fog corruption dataset. We observe that either adversarial snow or fog can effectively attack all TTA approaches with at least 60% of attack success rate.

Table 9. Average attack success rate of adversarially optimized snow corruptions and fog corruptions. [N_m=20; Targeted Attack] (Extended version of Table 4)

Dataset	Corruption	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
ImageNet-C	Snow	64.00	68.00	68.00	60.00	68.00	60.00
(ResNet-50)	Fog	84.00	76.00	72.00	76.00	76.00	76.00

D.7. Effectiveness of DIA under Relaxed Assumptions

In this subsection, we present the effectiveness of DIA under two relaxed assumptions of the attacker's capability.

DIA is still effective even if the attacker has no access to benign data and the batch is randomly selected (Table 10). We observe that the attack success rate of DIA is still high (e.g., 100% for 20 malicious samples) even if the attacker has no access to benign data or considers the random batch selection (RBS).

Table 10. Attack success rate with two relaxed assumptions of attacker's capability (i.e., access/no access to benign data and whether the random batch selection (RBS) is considered). [ImageNet; Targeted Attack] (Extended version of Table 5)

N_m	Benign Data	RBS	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
5 (2.5%)	Access	✗	92.00	92.00	72.00	56.0	64.0	52.0
	No access	✗	64.00	52.00	44.00	32.0	36.0	28.0
	No access	✓	60.00	44.00	40.00	28.0	32.0	28.0
10 (5%)	Access	✗	100.0	100.0	100.0	96.0	100.0	96.0
	No access	✗	100.0	92.0	84.0	88.0	88.0	84.0
	No access	✓	100.0	92.0	84.0	80.0	88.0	80.0
20 (10%)	Access	✗	100.0	100.0	100.0	100.0	100.0	100.0
	No access	✗	100.0	100.0	100.0	100.0	100.0	100.0
	No access	✓	100.0	100.0	100.0	100.0	100.0	100.0

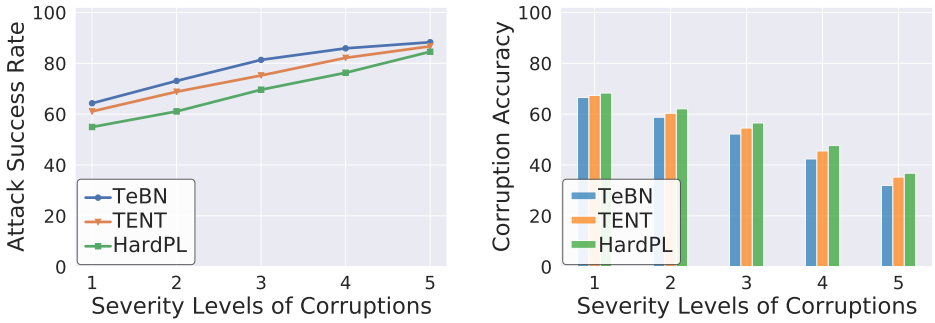
D.8. Effectiveness of DIA across Batch Sizes

We conduct additional experiments with batch sizes of 10, 20, 40, 80, 160, and 200 and a fixed 10% malicious samples rate. Our results, shown in the Table 11, indicate the attack success rate (ASR) remains similar for CIFAR-10-C and ImageNet-C (e.g., 55% for CIFAR-10-C and 100% for ImageNet-C). The results on CIFAR-100-C demonstrate a lower ASR for smaller batch sizes, but the ASR still remains highly effective. For instance, the ASR of **TeBN** drops to 75% with a batch size of 10.

Table 11. Average attack success rate with different batch size and fixed 10% malicious samples rate. [Targeted Attack]

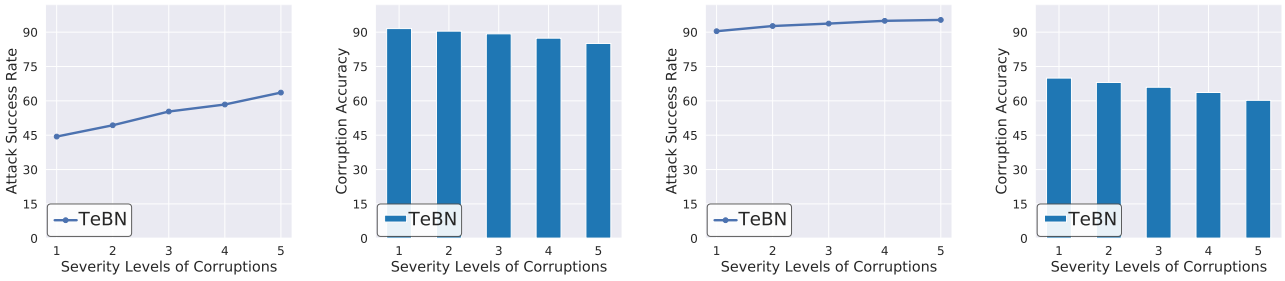
Dataset	TTA Methods	N_m=10	N_m=20	N_m=40	N_m=80	N_m=160	N_m=200
CIFAR-10-C (ResNet-26)	TeBN	60.84	55.11	56.72	57.97	57.88	55.47
	TENT	55.81	50.67	52.16	48.05	48.89	45.73
	HardPL	57.03	51.75	53.15	50.24	49.52	48.13
CIFAR-100-C (ResNet-26)	TeBN	75.25	80.51	86.72	90.03	88.68	93.73
	TENT	62.70	70.56	78.29	70.88	70.16	72.80
	HardPL	68.67	74.97	81.52	78.61	81.48	87.33
ImageNet-C (ResNet-50)	TeBN	100.0	99.97	100.0	100.0	100.0	100.0
	TENT	99.09	99.92	100.0	100.0	100.0	100.0
	HardPL	99.81	99.95	100.0	100.0	100.0	100.0

D.9. Effectiveness of DIA across Severities of Corruption



(a) Attack Success Rate (b) Corruption Accuracy

Figure 13. Illustration of how severity levels of corruption affect the corruption accuracy and attack success rate of our attack on the ImageNet-C dataset. (Line plot: attack success rate; bar plot: corruption accuracy) [N_m=5; Targeted Attack]



(a) CIFAR-10-C (b) CIFAR-100-C

Figure 14. Illustration of how severity levels of corruption affect the corruption accuracy and attack success rate of our attack on the CIFAR-C dataset. (Line plot: attack success rate; bar plot: corruption accuracy) [N_m=20; Targeted Attack]

Larger corruption severity tends to be more vulnerable against Distribution Invading Attack (Figure 13 and Figure 14). Our previous evaluation only concentrates on the level 3 severity (medium) of corruption; now, we analyze the impact of corruption severity. First, the accuracy significantly drops when corruptions are more severe, e.g., the average degradation from level 1 to level 5 is nearly 30% on ImageNet-C. Therefore, as expected, the ASR of the Distribution Invading Attack gets higher as the models (updated by TTA methods) tend not to be confident in their predictions. Similar behaviors are observed for the CIFAR-C dataset.

D.10. Effectiveness of DIA with Larger TTA Learning Rate

We conduct ablation studies (shown in Table 12) on the TTA learning rate η (set as 0.001 for previous experiments). Concretely, we increase η to 0.005 and evaluate DIA. Our single-level optimization method exhibits some performance degradations when the TTA learning rate rises but still reaches near-100% ASR with 20 (10%) malicious data. Furthermore, we also observe the corruption accuracy with $\eta = 0.005$ drop $\sim$4% on average.

Table 12. Effectiveness of Distribution Invading Attack with larger TTA learning rate (η) on ImageNet-C dataset.

N_m	η	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
5 (2.5%)	0.001	80.80	75.73	69.87	62.67	66.40	57.87
	0.005	80.80	52.00	46.93	35.73	45.33	32.00
10 (5%)	0.001	99.47	98.67	96.53	94.13	96.00	92.80
	0.005	99.47	87.73	83.73	74.67	82.13	71.47
20 (5%)	0.001	100.00	100.00	100.00	100.00	100.00	100.00
	0.005	100.00	100.00	99.73	98.13	100.00	98.67

D.11. Effectiveness of DIA with More TTA Optimization Steps

In this subsection, we conduct ablation studies (shown in Table 13) where the TTA optimization step is no longer 1 step but 5 steps. We observe a degradation (0% to 25%) in DIA performance if the TTA optimizes the unsupervised loss with 5 steps. However, with 20 malicious data, the ASR is still near-100%.

Table 13. Effectiveness of Distribution Invading Attack with more TTA optimization steps on ImageNet-C dataset.

N_m	Steps	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
5 (2.5%)	1	80.80	75.73	69.87	62.67	66.40	57.87
	5	80.80	58.13	53.07	39.73	49.87	36.00
10 (5%)	1	99.47	98.67	96.53	94.13	96.00	92.80
	5	99.47	93.60	88.53	79.73	87.20	77.87
20 (5%)	1	100.00	100.00	100.00	100.00	100.00	100.00
	5	100.00	100.00	100.00	99.73	100.00	99.47

D.12. Effectiveness of DIA with Additional Baseline Methods

D.12.1. Efficient Test-time Adaptation (ETA)

In this subsection, we present our DIA can also attack advanced methods of **TENT**, like **ETA** (Niu et al., 2022). In general, **ETA** introduces a technique to boost the efficiency of **TENT** by actively selecting reliable samples for updating the base model. However, they still suffer from the vulnerabilities of re-estimating BN statistics. Therefore, we can directly apply DIA, and our results demonstrate that **ETA is even more vulnerable than TENT for the CIFAR-C dataset (Table 14).** Furthermore, 20 malicious data can still achieve near-100% ASR on the ImageNet-C benchmark.

Table 14. Attack success rate of Distribution Invading Attack (targeted attacks) across benchmarks and TTA methods.

Dataset	N_m	TeBN(%)	TENT(%)	ETA(%)
CIFAR-10-C (ResNet26)	10 (5%)	25.87	23.20	24.53
	20 (10%)	55.47	45.73	53.07
	40 (20%)	92.80	83.87	89.87
CIFAR-100-C (ResNet26)	10 (5%)	46.80	26.40	33.60
	20 (10%)	93.73	72.80	86.67
	40 (20%)	100.00	100.00	100.00
ImageNet-C (ResNet50)	5 (2.5%)	80.80	75.73	41.33
	10 (5%)	99.47	98.67	81.60
	20 (10%)	100.00	100.00	99.47

D.12.2. Defensive Entropy Minimization (Dent)

Recently, Chen et al. (2022b) evaluated some previous defenses against conventional ℓ_p adversarial attacks that leverage transductive learning. We select Defensive Entropy Minimization (**Dent**) (Wang et al., 2021a) to defend against DIA as it is the most recent proposal. We conduct targeted attack experiments on ImageNet-C dataset with the standard and adversarially trained models, comparing **TeBN** (non-robust) and **Dent** with 20 and 40 malicious data. We observe that **Dent** provides some mitigations compared to TeBN, but **DIA still achieved over 60% attack success rate with 40(20%) malicious data.**

Table 15. Effectiveness of Distribution Invading Attack against Dent on ImageNet-C benchmark.

N_m	Model	TeBN(%)	Dent(%)	N_m	Model	TeBN(%)	Dent(%)
20 (10%)	Standard	100.0	77.80	**40 (20%)**	Standard	100.0	98.40
	Salman et al.	52.80	35.47		Salman et al.	92.50	67.37
	Engstrom et al.	44.20	36.53		Engstrom et al.	77.60	72.00

D.12.3. Test-time Training(TTT)

We also conduct experiments to evaluate whether Test-time Training(**TTT**) (Sun et al., 2020) is vulnerable under our threat model in this subsection. **TTT** leverages an auxiliary rotation prediction task at test time to improve prediction, which significantly differs from the methods we explored. Thus, we consider a simple attack by injecting rotated images as malicious data to confuse the auxiliary task and cause the model to update incorrectly. We perform the indiscriminate attack, which is to degrade the performance of all benign samples (excluding the malicious samples we injected).

To stay consistent with our setting in Section 5.3.1 , we assume that the test data for each trial contains 200 samples with 10, 20, and 40 of malicious data that is rotated 180 degrees and randomly injected. The model we used is a ResNet with GroupNorm(GN), the same as (Sun et al., 2020) , and the dataset is CIFAR-10-C. Table 16 report the benign error rate (the ratio of benign samples that is incorrectly classified by the model). We notice that this **simple attack causes the error rate on benign samples to rise** (e.g., from 14% to 25% for 40 malicious data). It is worth noting that, as we discussed in section 7, DIA may not be effective for all and future TTA methods. Our paper is to uncover that the TTA pipelines introduce yet another attack surface that can potentially be exploited by malicious parties.

Table 16. Effectiveness of Distribution Invading Attack aginst Test-time Training(TTT) on ImageNet-C dataset.

N_m	Model	No Attack(%)	Rotation Attack(%)
10(5%)	ResNet+GN	14.48	18.84
20(10%)	ResNet+GN	14.50	21.03
40(20%)	ResNet+GN	14.50	25.39

D.13. Effectiveness of DIA across Various Corruption Types

We then report the result of DIA for all 15 corruption types on Table 17. We select $N_m = 20$ for CIFAR-10-C and CIFAR-100-C and $N_m = 5$ for ImageNet-C. We observe that the predictions under Brightness corruption are the hardest to attack. This is the expected result because the benign accuracy of Brightness is the highest (Schneider et al., 2020).

Table 17. Effectiveness of Distribution Invading Attack across various corruption types. (CIFAR-C: N_m=20; ImageNet-C: N_m=5)

Dataset	Corruption Type	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
CIFAR-10-C (ResNet-26) (N_m=20 (10%))	Gaussian Noise	68.0	48.0	52.0	50.0	56.0	50.0
	Shot Noise	70.0	54.0	56.0	62.0	62.0	54.0
	Impulse Noise	76.0	48.0	58.0	54.0	62.0	50.0
	Defocus Blur	44.0	38.0	38.0	40.0	40.0	40.0
	Glass Blur	50.0	44.0	52.0	44.0	48.0	42.0
	Motion Blur	54.0	48.0	46.0	50.0	52.0	46.0
	Zoom Blur	44.0	46.0	44.0	46.0	44.0	44.0
	Snow	58.0	44.0	42.0	44.0	50.0	44.0
	Frost	48.0	46.0	48.0	50.0	48.0	48.0
	Fog	60.0	40.0	46.0	44.0	48.0	40.0
	Brightness	38.0	38.0	40.0	38.0	38.0	40.0
	Contrast	62.0	48.0	48.0	48.0	50.0	50.0
	Elastic Transform	56.0	56.0	58.0	50.0	54.0	52.0
	Pixelate	44.0	46.0	46.0	44.0	40.0	44.0
	JPEG Compression	60.0	42.0	48.0	48.0	50.0	42.0
	All	55.47	45.73	48.13	47.47	49.47	45.73
CIFAR-100-C (ResNet-26) (N_m=20 (10%))	Gaussian Noise	100.0	80.0	92.0	86.0	94.0	74.0
	Shot Noise	100.0	80.0	98.0	82.0	86.0	82.0
	Impulse Noise	96.0	82.0	92.0	90.0	90.0	74.0
	Defocus Blur	86.0	64.0	82.0	72.0	82.0	68.0
	Glass Blur	96.0	78.0	80.0	78.0	78.0	66.0
	Motion Blur	94.0	74.0	82.0	80.0	82.0	74.0
	Zoom Blur	90.0	60.0	80.0	62.0	70.0	60.0
	Snow	92.0	72.0	80.0	74.0	72.0	66.0
	Frost	94.0	72.0	94.0	74.0	82.0	78.0
	Fog	94.0	76.0	92.0	86.0	92.0	76.0
	Brightness	86.0	64.0	84.0	72.0	72.0	68.0
	Contrast	96.0	80.0	94.0	88.0	94.0	78.0
	Elastic Transform	100.0	74.0	90.0	80.0	84.0	70.0
	Pixelate	90.0	64.0	84.0	80.0	78.0	72.0
	JPEG Compression	92.0	72.0	86.0	74.0	88.0	68.0
	All	93.73	72.80	87.33	78.53	82.93	71.60
ImageNet-C (ResNet-50) (N_m=5 (2.5%))	Gaussian Noise	92.0	92.0	72.0	56.0	64.0	52.0
	Shot Noise	88.0	76.0	68.0	56.0	60.0	48.0
	Impulse Noise	92.0	84.0	80.0	68.0	80.0	60.0
	Defocus Blur	100.0	96.0	88.0	84.0	84.0	80.0
	Glass Blur	92.0	84.0	84.0	68.0	72.0	64.0
	Motion Blur	88.0	88.0	84.0	68.0	76.0	68.0
	Zoom Blur	84.0	76.0	76.0	72.0	76.0	68.0
	Snow	72.0	64.0	56.0	56.0	56.0	56.0
	Frost	88.0	72.0	72.0	64.0	68.0	52.0
	Fog	88.0	84.0	80.0	76.0	80.0	68.0
	Brightness	48.0	48.0	44.0	40.0	40.0	40.0
	Contrast	96.0	96.0	88.0	84.0	88.0	80.0
	Elastic Transform	56.0	60.0	52.0	48.0	48.0	40.0
	Pixelate	56.0	52.0	52.0	44.0	48.0	36.0
	JPEG Compression	72.0	64.0	52.0	56.0	56.0	56.0
	All	80.80	75.73	69.87	62.67	66.40	57.87

D.14. Results of Greedy Model Space Attack (GSMA)

We also evaluate the another optimization method GSMA-MIN (instead of PGD) (Chen et al., 2022b) on the CIFAR-10-C and CIFAR-100-C benchmark, with 10, 20, and 40 malicious data. We modified the objective of GSMA-MIN to attack a single benign input (DIA targeted attack) and maintained the default attack ensemble size of $T = 3$. We compared GSMA-MIN with DIA via PGD, which serves as the default method in our paper. The attack success rates are presented in Table 18, and we observed **nearly identical performance** between the two methods.

Table 18. Attack success rate of Distribution Invading Attack with **GMSA**.

Dataset	N_m	Optimization	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
CIFAR-10-C (ResNet-26)	10 (5%)	PGD	25.87	23.20	25.33	23.20	24.80	23.60
	10 (5%)	GMSA-MIN	26.27	24.40	24.40	23.47	26.13	23.60
	20 (10%)	PGD	55.47	45.73	48.13	47.47	49.47	45.73
	20 (10%)	GMSA-MIN	55.07	45.47	48.93	47.73	49.33	45.73
	40 (10%)	PGD	92.80	83.87	84.27	82.93	86.93	85.47
	40 (10%)	GMSA-MIN	92.93	83.87	84.40	82.93	87.07	84.93
CIFAR-100-C (ResNet-26)	10 (5%)	PGD	46.80	26.40	31.20	27.60	32.13	26.13
	10 (5%)	GMSA-MIN	47.47	26.80	31.07	27.87	32.27	25.60
	20 (10%)	PGD	93.73	72.80	87.33	78.53	82.93	71.60
	20 (10%)	GMSA-MIN	94.27	72.80	87.07	78.13	84.40	71.33
	40 (10%)	PGD	100.00	100.00	99.87	100.00	99.87	100.00
	40 (10%)	GMSA-MIN	100.00	100.00	100.00	100.00	99.87	100.00

E. Additional Experiments on Robust Models and Mitigation Methods

In this appendix, we show some additional findings and results of our mitigating methods as the complement of Section 6.

The roadmap of this section is as follows. We first investigate the role of robust models from two aspects. In Appendix E.1, we apply TTA to the adversarially trained models and surprisingly witness an improvement in clean accuracy. We then evaluate the robust models' performance facing the corruption accuracy in Appendix E.2 and analyze their effectiveness in resisting DIA attacks in Appendix E.3. Furthermore, we extensively study the Robust Batch Normalization (BN) estimation. In Appendix E.4, we visualize the behaviors of Layer-wise BN. Then we provide the additional results of Robust BN Estimation on CIFAR-C in Appendix E.5, ImageNet-C in Appendix E.6. Finally, we conduct a parameter search for Robust BN estimation in Appendix E.7.

E.1. Additional Findings of Adversarial Models and TTA on Clean Data

Dataset	Models	Source(%)	TeBN(%)	TENT(%)	Hard PL(%)	Soft PL(%)	Robust PL(%)	Conjugate PL(%)
CIFAR-10	(Dai et al., 2022) WRN-28	87.02	87.00	**91.74**	90.83	91.62	90.93	91.68
	(Wu et al., 2020a) WRN-28	88.25	86.35	92.81	91.35	**93.07**	93.02	91.59
	(Gowal et al., 2021) WRN-28	87.49	87.42	88.39	88.22	**88.44**	88.21	**88.44**
	(Sehwag et al., 2022) RN-18	84.59	84.38	87.24	87.25	**87.63**	87.05	86.73
CIFAR-100	(Sehwag et al., 2022) WRN-34	65.93	67.37	73.96	70.63	**74.06**	73.93	72.81
	(Wu et al., 2020a) WRN-34	60.38	55.37	62.01	59.43	**62.19**	61.89	61.87
	(Chen et al., 2022a) WRN-34	64.07	61.94	67.58	61.29	68.11	**69.09**	65.02
	(Jia et al., 2022) WRN-34-20	67.31	64.17	69.72	68.60	**70.21**	69.97	69.71

Table 19. Clean accuracy can be improved by the test-time adaptation for adversarially trained models on CIFAR-10 and CIFAR-100. As a reference, the accuracy of the standard Wide ResNet with 28 layers (WRN-28) is 94.78% for CIFAR-10 and 78.78% for CIFAR-100. (RN-18 is ResNet-18, WRN-34 is Wide ResNet 34)

One of the notable weaknesses of adversarial training is the apparent performance degradation on clean test data (Madry et al., 2018; Xie et al., 2019), owing to the divergent distributions of adversarial and clean inputs for batch normalization (BN). Therefore, (Xie et al., 2020) proposed to use distinct BN layers to handle data from two distributions and (Wang et al., 2022a) leveraged normalizer-free networks (removing BN layers) to improve clean accuracy. As we discussed, TTA methods exhibit promising improvement in mitigating distribution shifts. Therefore, we consider if they can improve the accuracy of clean input when training data draws from the distribution of adversarial data.

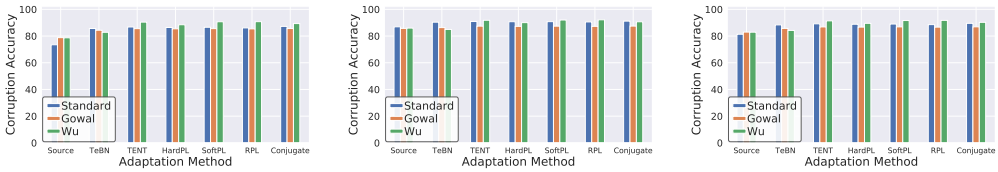
TTA methods achieve a non-trivial clean accuracy improvement for adversarially trained models on CIFAR-10 and CIFAR-100 (Table 19). We select a list of adversarial training methods (Dai et al., 2022; Wu et al., 2020a; Gowal et al., 2021; Sehwag et al., 2022; Chen et al., 2022a; Jia et al., 2022) as the source model and use TTA methods on clean inputs. All of them are downloaded from RobustBench (Croce et al., 2020). As shown, **Soft PL** significantly improves the average accuracy $\sim$3.35% for CIFAR-10 and $\sim$4.22% for CIFAR-100. The most significant improvement even boosts 8.13% compared to no TTA method. This result is interesting as another perspective to understand TTA better, and we hope future works can explore more on it. We also observe that simply replacing the BN statistics does not work as well as using adversarial models on ImageNet.

E.2. Corruption Accuracy of Robust Models

We evaluate the corruption accuracy, where Wide ResNet with 28 layers trained by Gowal et al. (2021) and Wu et al. (2020a) are chosen for CIFAR-10-C, and Wide ResNet with 28 layers developed by Pang et al. (2022) and Rebuffi et al. (2021) are selected for CIFAR-100-C. Note that all adversarial models are downloaded from RobustBench (Croce et al., 2020).

Robust models with TTA achieve better performance than the standard model on CIFAR-10-C, but worse performance on CIFAR-100-C (Figure 15 and Figure 16). For CIFAR-10-C, we observe a performance boost from using robust models. For example, Wu et al. (2020a) achieves a $\sim$5% improvement than the standard one. For CIFAR-100-C, robust models cannot perform similarly to the standard model. However, leveraging adversarial models and TTA can still exceed the standard model without TTA. Since they may use different training configurations (e.g., training batch size, data augmentations, inner optimization steps), we leave the question of "training the best adversarial source model for TTA" as future works.

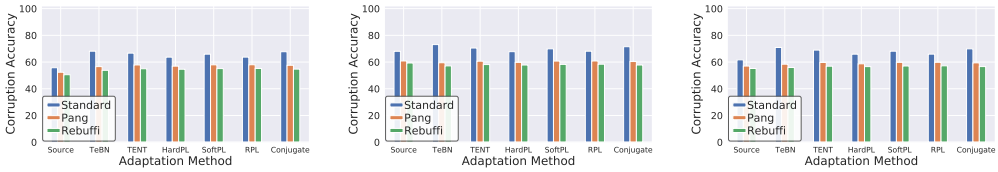
Robust models achieve comparable performance with the standard model on ImageNet-C if TTA is deployed (Figure 17). We then evaluate the corruption accuracy of robust models on the ImageNet-C dataset. Specifically, adversarially trained ResNet 50 models originated by Salman et al. (2020) and Engstrom et al. (2019) are selected. We observe that the average corruption accuracy of adversarial models can significantly benefit from test-time adaptation methods. For example, the average corruption accuracy of robust models is $\sim$10% less than the standard model but reaches similar performance when TTA is deployed. Furthermore, robust models generally gain higher accuracy on larger severity than the standard ones (e.g., $\sim$10% improvement on severity 5 with TTA).



(a) Average severity (b) Severity 1 (c) Severity 2

(d) Severity 3 (e) Severity 4 (f) Severity 5

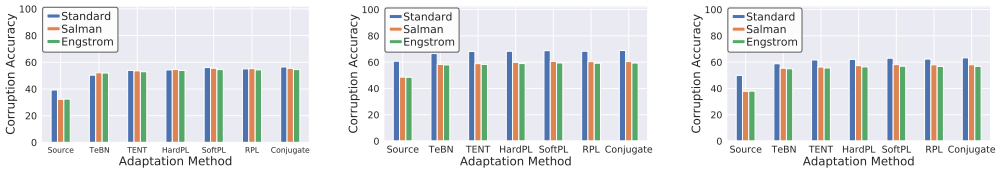
Figure 15. Corruption accuracy of the standard and robust models (Gowal et al. (2021) and Wu et al. (2020a)) on CIFAR-10-C under different severity levels.



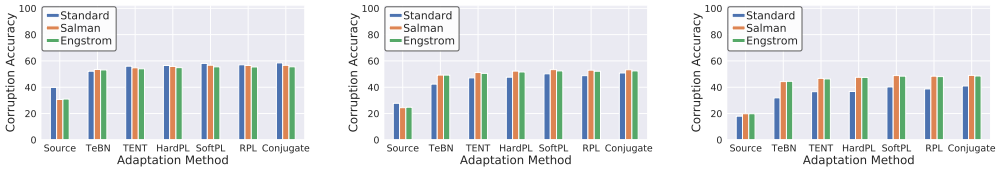
(a) Average severity (b) Severity 1 (c) Severity 2

(d) Severity 3 (e) Severity 4 (f) Severity 5

Figure 16. Corruption accuracy of the standard and robust models (Pang et al. (2022) and Rebuffi et al. (2021)) on CIFAR-100-C under different severity levels.



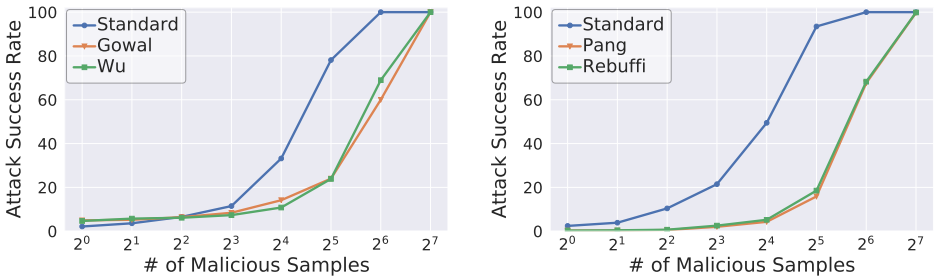
(a) Average severity (b) Severity 1 (c) Severity 2



(d) Severity 3 (e) Severity 4 (f) Severity 5

Figure 17. Corruption accuracy of the standard and robust models (Salman et al. (2020) and Engstrom et al. (2019)) on ImageNet-C under different severity levels.

E.3. Mitigating Distribution Invading Attacks by Robust Models on CIFAR-C Benchmarks



(a) CIFAR-10-C (b) CIFAR-100-C

Figure 18. Attack success rate of our proposed attacks against robust models across the numbers of malicious samples. As a reference, 128 is 64% of the whole batch containing 200 samples. Robust models for CIFAR-10-C are Gowal et al. (2021) and Wu et al. (2020a); robust models for CIFAR-100-C are Pang et al. (2022) and Rebuffi et al. (2021). [TTA method: **TeBN**]

In Figure 18, we report the ASR of standard and two robust models across the number of malicious samples on the CIFAR-C dataset. **Adversarial models mitigate the vulnerability against DIA but cannot fully alleviate it.** For example, with 32 malicious samples, robust models degrade $\sim$60% and $\sim$70% ASR for CIFAR-10-C and CIFAR-100-C, respectively. However, increasing malicious samples still breaks robust source models.

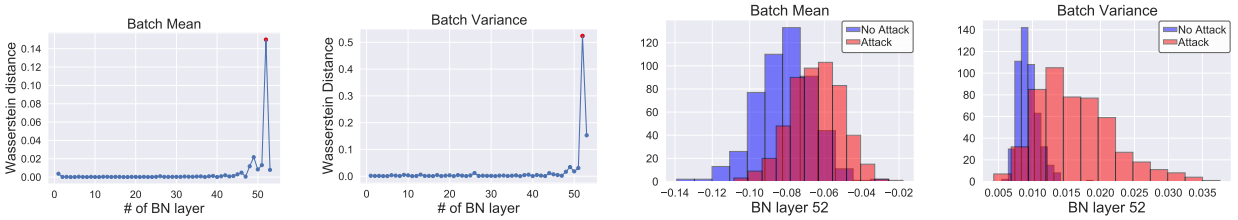
E.4. Understanding the Layer-wise Batch Normalization behaviors

To further design a more robust approach, we seek to understand what happens to BN statistics when deploying the Distribution Invading Attack. Therefore, we visualize the BN mean histogram and BN variance histogram of all 53 layers in Figure 29 and 30, with and without attacks. We use the *Wasserstein distance* with ℓ_1-norm to measure the discrepancy between two histograms. Specifically, given two probability measures μ_w, μ_v over $\mathbb{R}$, the Wasserstein distance under Kantorovich formulation (Kantorovich, 1942) is defined as

$$D_{\mathrm{W}}(\mu_w, \mu_v) := \min_{\pi \in \Pi(\mu_w, \mu_v)} \int_{\mathbb{R}\times\mathbb{R}} |z - z'| d\pi(z, z') \tag{17}$$

where $\Pi(\mu_w, \mu_v) := \left\{\pi \in \mathcal{P}(\mathbb{R}\times\mathbb{R}) \mid \int_{\mathbb{R}} \pi(z, z')dz = \mu_w, \int_{\mathbb{R}} \pi(z, z')dz' = \mu_v\right\}$ denotes a collection of couplings between two distributions μ_w and μ_v. Here, we view each histogram as a discrete probability distribution in a natural sense. Note that the histograms for different layers may have different overall scales; hence, we normalize the ℓ_1 distance $|z - z'|$ by the range of the histogram of benign samples for a fair comparison.

Figure 19(a) presents the Wasserstein distance for each layer, and interestingly, we observe that D_{W} remains low on most layers but leaps on the last few layers, especially the BN layer 52. Then, we visualize the histograms of layer 52 in Figure 19(b), which appears to be an obvious distribution shift.

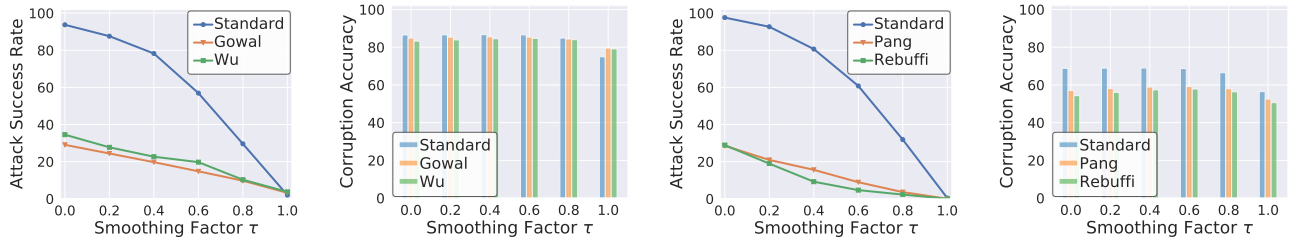


(a) Wasserstein Distance (b) BN Statistics Histogram

Figure 19. (a) Wasserstein Distance between malicious and benign BN Statistics gets unexpectedly high at BN Layer 52. (b) Histogram of batch mean and variance at BN Layer 52. (More figures are presented in Figure 29 and Figure 30).

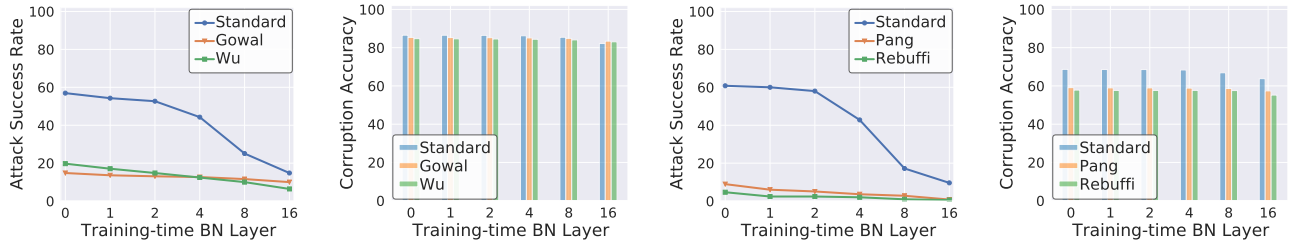
E.5. Effectiveness of Robust BN Estimation on CIFAR-C

We report the effectiveness of our two robust estimation methods, smoothing via training-time BN statistics and adaptively selecting layer-wise BN statistics on CIFAR-C. Precisely, we use $N_m = 40$, which is very challenging for defense (ASR reaches near-100%). Again, we select $\tau = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ for balancing training-time and test-time BN. For adaptively selecting layer-wise BN statistics, we leverage full training-time BN (τ = 1.0) for the last N_{tr} = {0, 1, 2, 4, 8, 16} BN layers.



(a) CIFAR-10-C (b) CIFAR-100-C

Figure 20. Controlling τ = 0.6 can degrade the attack success rate (line) while maintaining high corruption accuracy (bar). [N_m=40]



(a) CIFAR-10-C (b) CIFAR-100-C

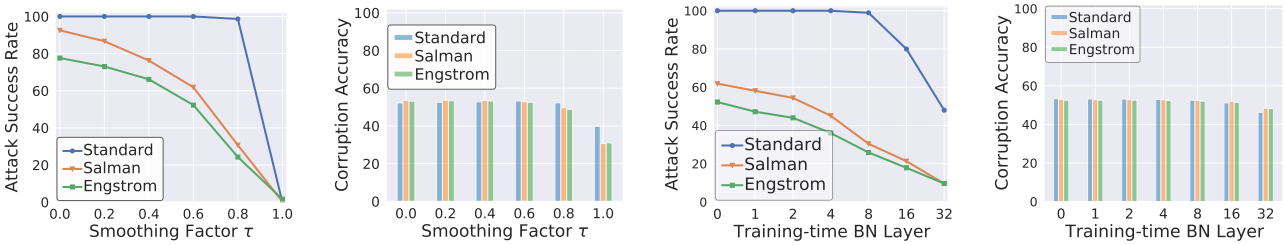
Figure 21. Controlling layer-wise BN can degrade the attack success rate (line) while maintaining high corruption accuracy (bar). [N_m=40; τ=0.6]

Robustly estimating the final BN statistics significantly enhance performance against DIA. Figure 20 demonstrates the effectiveness of smoothing with training-time BN, which degrades the ASR ∼30% for the standard model if we select τ = 0.6. Then, we stick with τ = 0.6 and apply layer-wise BN. As a result, the ASR further degrades by about 15% without

sacrificing the corruption accuracy (shown in Figure 21). In conclusion, applying two of our robust estimating methods can degrade ∼45% for the standard model and ∼30% for the adversarial model in total.

Our method involves a trade-off between TTA performance and robustness to DIA attacks. It does not inherently eliminate vulnerability when exploiting the test batch information. In addition, selecting the right hyper-parameters (i.e., τ and N_{tr}) without the out-of-distribution information is another challenging problem.

E.6. Addtional Experiments of Robust BN Estimation on ImageNet-C



(a) Smoothing via training-time BN statistics (b) Adaptively selecting layer-wise BN statistics. [τ=0.6]

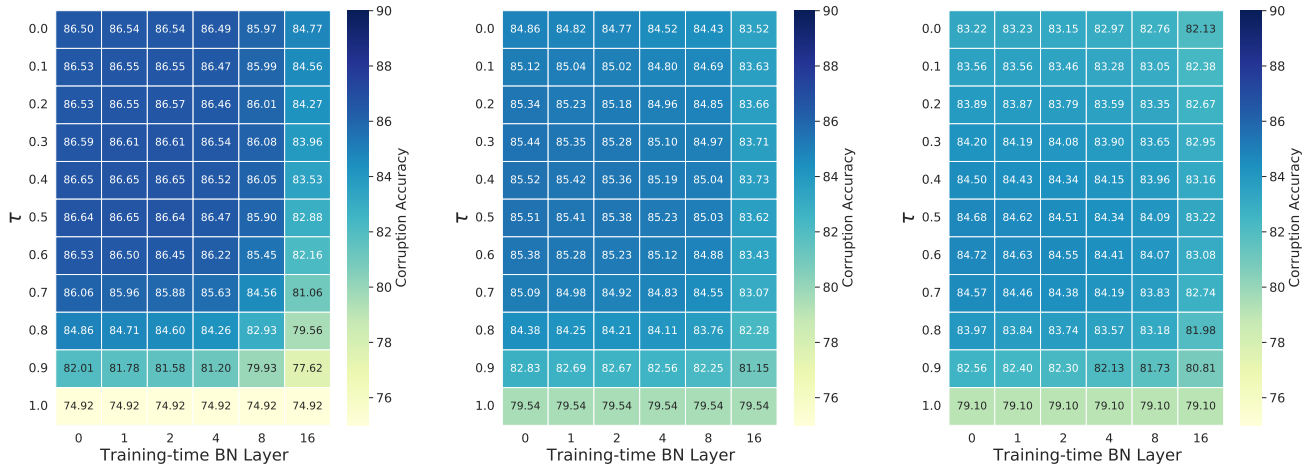
Figure 22. Robustly estimating the final BN statistics can degrade the attack success rate (line) while maintaining high corruption accuracy (bar) on ImageNet-C. [N_m=40] This is the complementary evaluation of Figure 7 and Figure 8 with more malicious samples.

Figure 22 shows the additional experiments of section 6.2 with a stronger attacking setting (N_m = 40). We observe the conclusions keep the same where selecting an appropriate τ and N_{tr} (i.e., τ = 0.6, N_{tr} = 16) can decrease the ASR for ∼70% and ∼20% for the robust and standard model, respectively, and maintain the corruption accuracy.

E.7. More Evaluations of Corruption Accuracy for Robustly Estimating BN

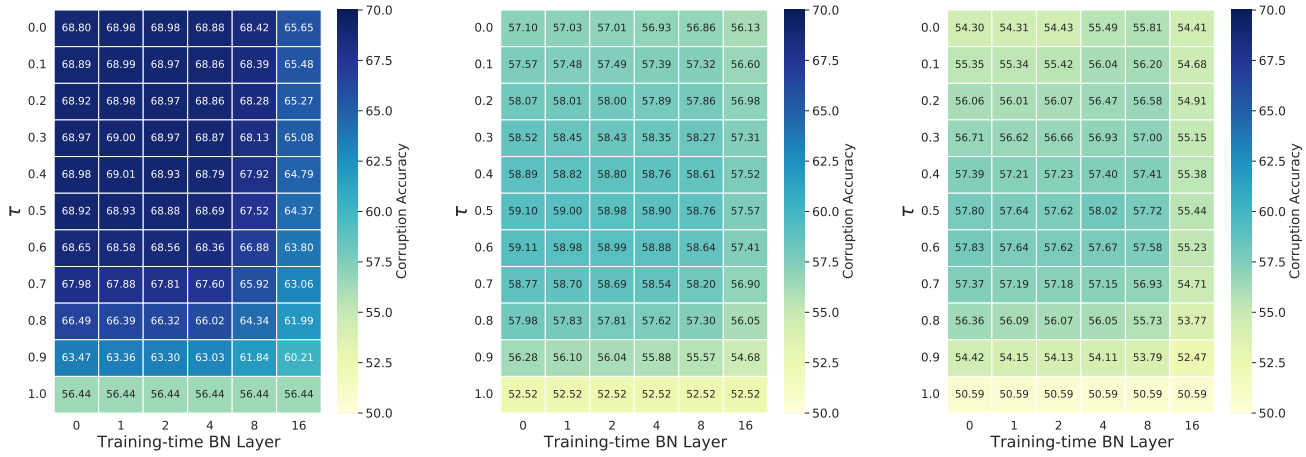
In this subsection, we present a more comprehensive parameters search for the smoothing factor τ and number of training-time BN layers N_{tr} to understand their effect on corruption accuracy. Specifically, we analyze standard and robust models on CIFAR-C and ImageNet-C benchmarks.

As shown in Figure 23, Figure 24, and Figure 25, we observe that the corruption accuracy generally reaches the best when $\tau = 0.5$ and $N_{tr} = 0$. The improvement compared to **TeBN** ($\tau = 0$ and $N_{tr} = 0$) is limited (∼2%) except for the robust models on CIFAR-100-C. The corruption accuracy drops dramatically for $\tau > 0.7$. Furthermore, increasing N_{tr} also decreases the accuracy of corrupted data.



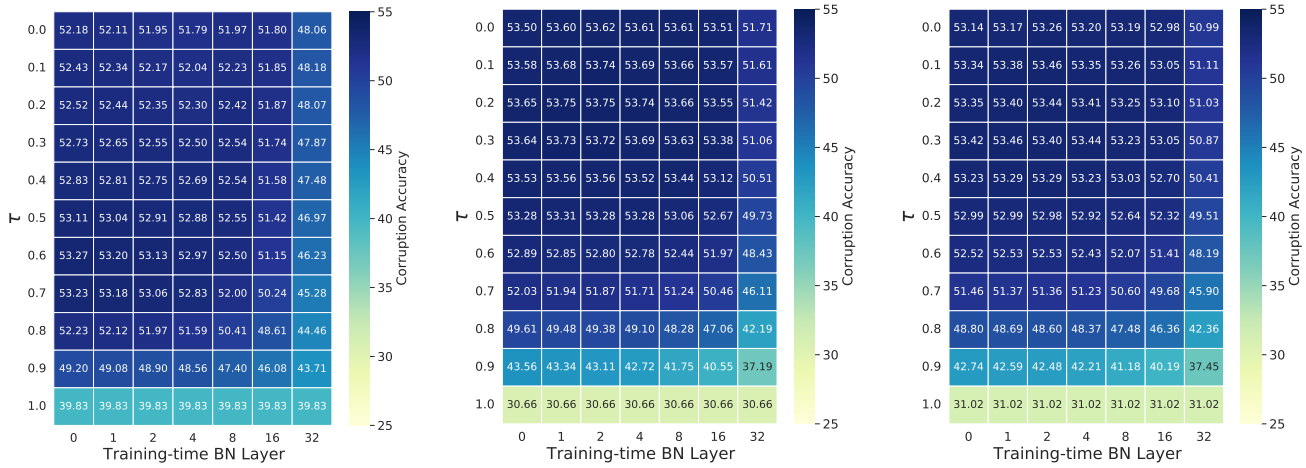
(a) Standard WRN-28 (b) Gowal et al. (2021) (c) Wu et al. (2020a)

Figure 23. Corruption accuracy of balancing training-time BN across τ and number of Training-time BN layers on CIFAR-10-C.



(a) Standard WRN-28 (b) Pang et al. (2022) (c) Rebuffi et al. (2021)

Figure 24. Corruption accuracy of balancing training-time BN across τ and number of Training-time BN layers on CIFAR-100-C.



(a) Standard ResNet-50 (b) Salman et al. (2020) (c) Engstrom et al. (2019)

Figure 25. Corruption accuracy of balancing training-time BN across τ and number of Training-time BN layers on ImageNet-C.

F. Visiualization of Constrained Attacks

F.1. Examples of Corrupted Images and Corresponding L_∞ Malicious Images

Figure 26. Demonstrations of 15 types of corrupted benign images (upper) and malicious images with $\epsilon = 8/255$ (lower) from the ImageNet-C benchmark. Most malicious images are visually imperceptible compared with their original ones. [Severity level: 3]

F.2. Examples of Corrupted Images and Adversarial Corrupted Images

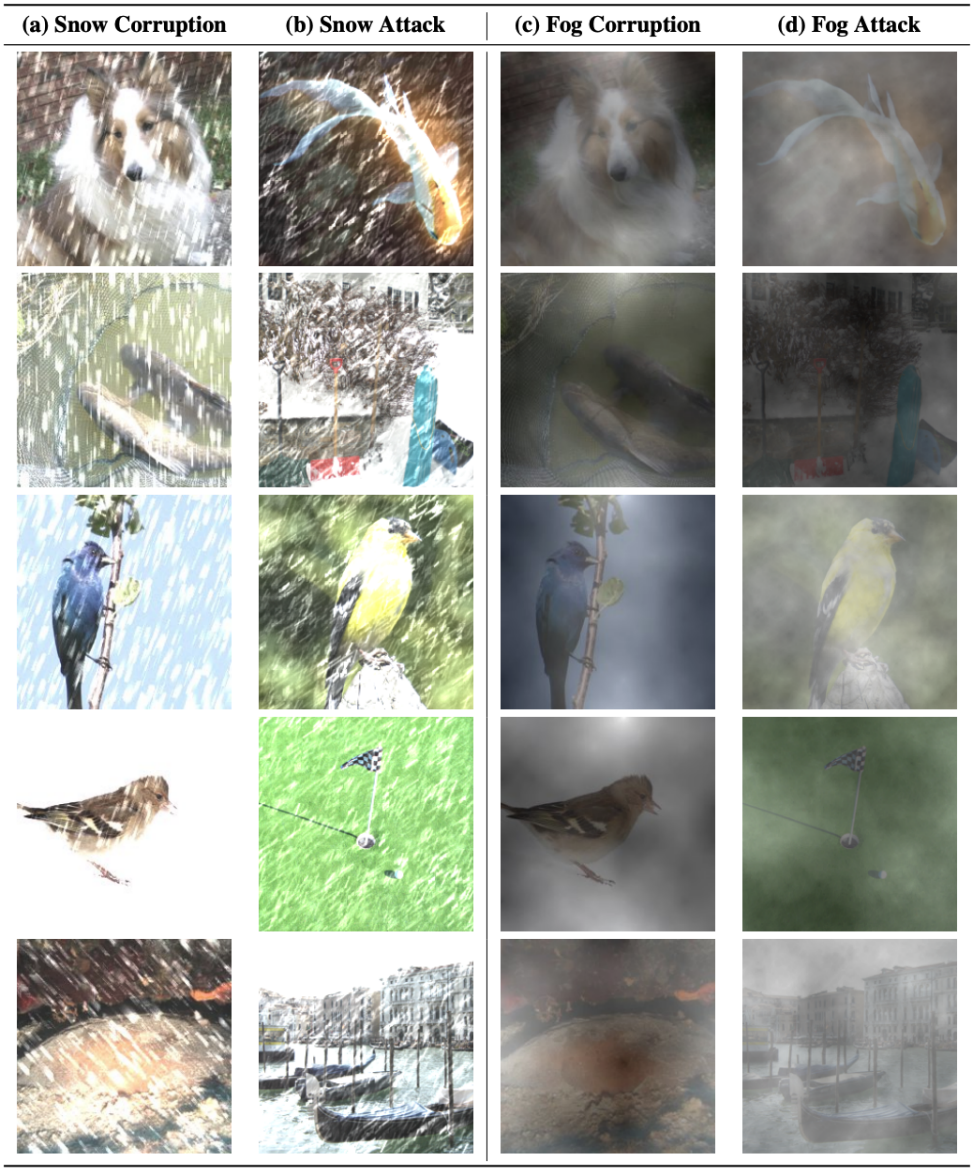


Figure 27. Demonstrations of the corrupted snow and fog dataset from ImageNet-C with severity level 3 and adversarially corrupted images (DIA). We observe that manually distinguishing malicious and benign samples is hard.

F.3. Examples of Distribution Invading Attack without Constraints

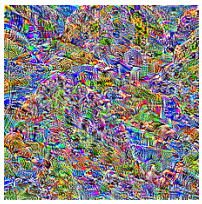


Figure 28. A demonstration of unconstrained Distribution Invading Attack. The resulting image is largely dependent on the initialization process, which uses random noise in this case.

G. Visualization of Batch Normalization Statistics

Figure 29. Full layer-by-layer visualization of BN mean histogram with and without DIA attacks. The distribution differs on the last few layers.

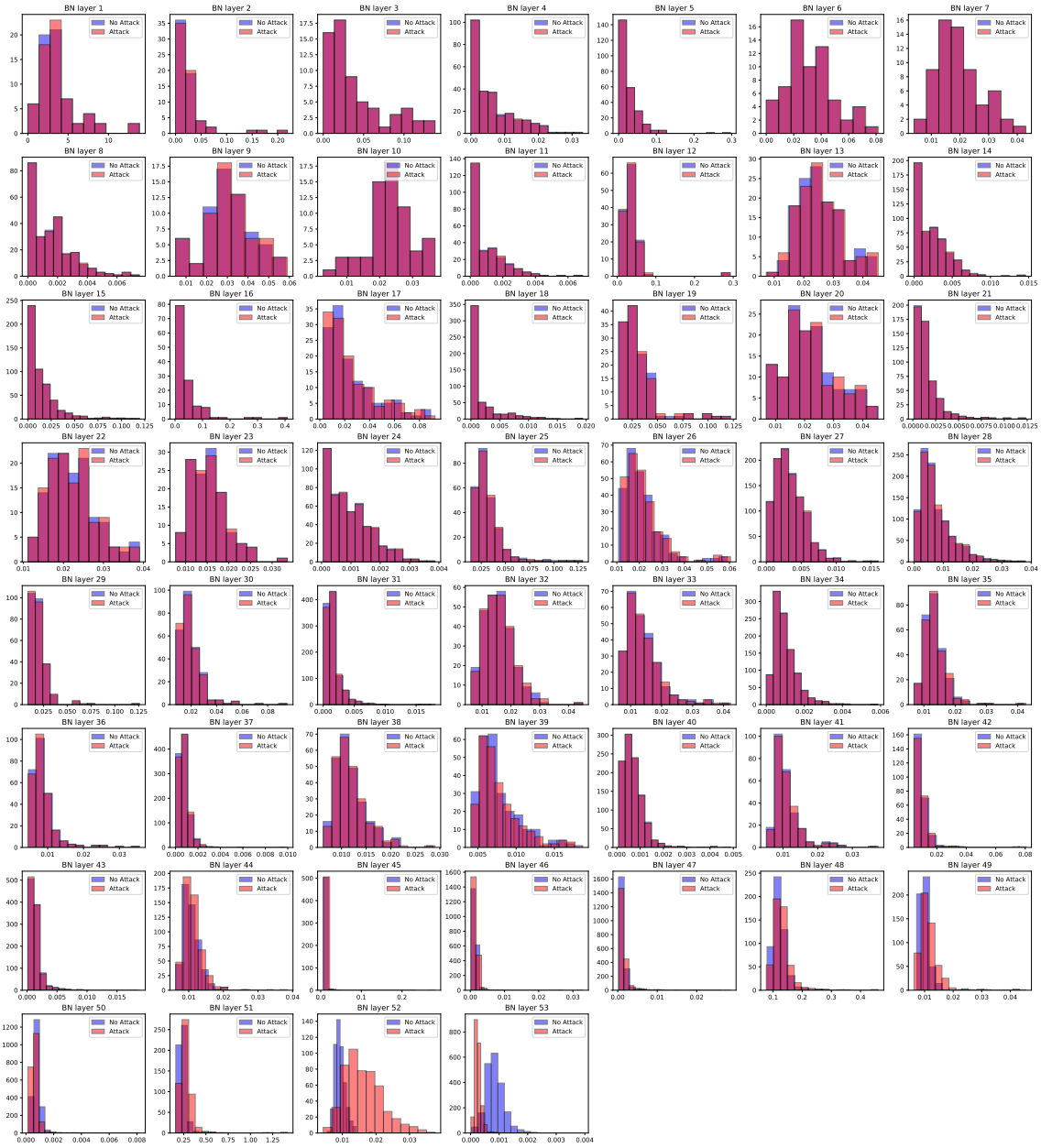


Figure 30. Full layer-by-layer visualization of BN variance histogram with and without DIA attacks. The distribution differs on the last few layers.