SITA: Single Image Test-time Adaptation

Ansh Khurana[1*], Sujoy Paul[2], Piyush Rai[3*], Soma Biswas[4], Gaurav Aggarwal[2]
[1]Stanford University, [2]Google Research, [3]IIT Kanpur, [4]IISc Bangalore

Abstract

In Test Time Adaptation (TTA), given a source model, the goal is to adapt it to make better predictions for test instances from a different distribution than the source. Crucially, TTA assumes no access to the source data or even any additional labeled/unlabeled samples from the target distribution. In this work, we consider TTA in a more pragmatic setting which we refer to as SITA (Single Image Test-time Adaptation). Here, when making a prediction, the model has access only to the given single *test instance, rather than a* batch *of instances, the latter being typically considered in the literature. This is motivated by the real scenarios where inference is needed on-demand instead of delaying for an incoming batch, or the inference is needed on an edge device (like mobile phones) where there is no scope for batching. The entire adaptation process in SITA should be extremely fast, as it happens at inference time. To address this, we propose a novel approach that requires only a single forward pass. There are two key components to our adaptation process: 1. estimating the normalization parameters from only a single test instance using label-preserving transformations, 2. calibrating the estimate for each test instance based on their distance from the source distribution. We perform these without any back-propagation steps, which makes our model much faster than recent test time adaptation methods. Our method can be used on any off-the-shelf trained model for both classification and segmentation tasks. Despite being very simple, our method is able to achieve significant performance gains compared to directly applying the source model on the target instances, as reflected in our extensive experiments and ablation studies along three dimensions - datasets, tasks, and network architectures.*

1. Introduction

Deep neural networks work remarkably well on a variety of applications, specifically when the test samples are drawn from the same distribution as the training data. The performance falls drastically when there is a non-trivial shift between the train and test distributions [7, 21]. In several real-world settings, such a performance drop may make the model unusable. There have been recent works in the literature to train robust models [9, 20, 25]. While this is a viable research direction for the problem at hand, it entails modifying the training process. This may not always be practical as the training data may no longer be available due to privacy/storage concerns. All that is available is a previously trained model. Therefore, there is a growing interest in *Test Time Adaptation* (TTA), where the models can be adapted at test time, without changing the training process or requiring access to the original training data.

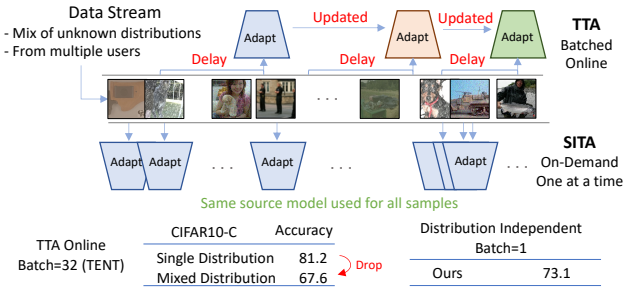


Figure 1. **SITA**: **S**ingle **I**mage **T**est-time **A**daptation. SITA is a realistic version of test-time adaptation setting. The model only has access to the given test instance, rather than a batch of instances. Online methods with large batch sizes only work well when encountering test data from a single distribution at a time, and suffer a considerable drop (81.2 to 67.6) in performance when images from different distributions are mixed together and sent as a stream. Contrastingly, in SITA, since adaptation is done for each individual test sample independently, it does not suffer from this problem.

TTT [28] and TENT [29] are among recent works that have been very effective in adapting models at prediction time. TTT [28] uses auxiliary self-supervised tasks to train the source model. The model is then fine-tuned (via the self-supervised sub-network) for every test instance for multiple iterations. Using TTT for adapting a new model, one needs to modify the training process (adding self-supervised sub-network) and hence needs access to the source training data, which may not be always available. Further, the multiple backward passes take considerable amount of time which may not be available where latency is not acceptable. TENT [29], on the other hand, adapts the given trained model, without accessing the source data. TENT assumes

*Work done while at Google Research.

that the data comes in batches, with batch size usually much greater than one. It considers an online setting, where the model adapted to the current instances (batch) is used for adapting the subsequent instances (batches), which implies that the model has information about all the test instances seen till a certain point.

For example, consider a recommendation model deployed on an edge-device that takes in a photograph of the user's surroundings as input. In this case, the user would provide the model with a single test image for inference and adaptation. The model does not have a large batch of samples for model adaptation, and the subsequent queries provided by the user at different time frames/locations would not belong to the same distribution for online adaptation, as the user's environment keeps changing. Thus, in most deployment scenarios, the assumptions made by TTT [28] and TENT [29] do not hold. Motivated by these, we enumerate the following desirable properties of algorithms developed for the realistic, and challenging SITA protocol:

- Does not require access to the source training data.
- Almost as fast as the original model during inference.
- Adapt to a single test instance without requiring a batch.
- Model adapted to a test instance should not be used on subsequent instances.
- Devoid of hyper parameter tuning at test time.

The first property is related to both privacy/storage issues and speed, as any re-training on the source data would make the approach much slower. The motivation behind the third property is latency and privacy. For large batch sizes, one has to wait for a certain number of samples, leading to delays, or club samples from multiple users, which may have privacy concerns. The fourth property is motivated by the fact that different test instances may come from very different distributions, which will adversely affect the performance of the model as highlighted in Figure 1. It shows that online methods which use large batch sizes, like TENT [29], improve performance by evaluating one corruption type at a time (single distribution), i.e., resetting the model for the next corruption type. When evaluated by mixing all 15 corruption types in the CIFAR-10-C [8] dataset (mixed distribution), their performance drops considerably. The last property is required as we cannot expect validation examples to tune hyperparameters at test-time for different target distributions.

In this work, we propose a method which overcomes the aforementioned limitations and fulfills the various desiderata of the **S**ingle **I**mage **T**est-time **A**daptation (SITA). The proposed method does not assume any access to the source data, adapts to one test instance at a time using only a single forward pass and resets the model to the given source model for adapting to every new test instance. Our method optimizes the batch-norm statistics of the network for every test instance. This involves two processes - (1) estimating the batch-norm statistics from only a single test image, which is a challenging problem [26], and (2) combining the estimated statistics with the source statistics by choosing an optimal parameter for every test instance which respects the fact that different test instances can be at different distances from the source distribution. Our adaptation method uses only one forward pass and thus is quite fast (comparable to using the source model directly), unlike other TTA methods in literature which require at least one backward pass. The proposed approach shows consistent performance boosts across a variety of datasets for both classification and segmentation, as well as for various network architectures.

The main contributions of this work are as follows:

1. We formalise the **S**ingle **I**mage **T**est-time **A**daptation (SITA) setting.
2. We propose a fast adaptation approach that performs instance-specific calibration, eliminating the need to tune hyper-parameters for different target distributions. Our approach is deployment-friendly as it operates with limited compute and runtime, and uses only a single forward pass for adaptation.
3. We achieve state-of-the-art performance for SITA across all tasks (segmentation and classification), datasets, and network architectures.

2. Related Work

Table 1 summarizes the characteristics of various settings which adapt a model trained on a *source* distribution to a *target* distribution. Fine-tuning and domain adaptation are offline methods, that is, they assume access to the entire source and target dataset to adapt and thus are out of scope for *test time* settings. We divide the discussion on related works based on the characteristics of the *test time* setting proposed by recent works.

Source-free Domain Adaptation. One of the constraints of SITA is that we cannot use the source data while adaptation, but only use the trained source model. Recent works in literature tackle this problem, but using an unlabeled target set for adaptation, with the hypothesis that the test instances are drawn from the target distribution. These methods include - entropy minimization with divergence maximization [15], pseudo-labeling with self-reconstruction [31], as well as generating additional target images [13, 16] and robustness to dropout [24]. Unlike these methods, in SITA, we do not have any target set to adapt and do not have any assumption over the distribution of the test instances.

Test Time Training. In this setting, self-supervised tasks are introduced during training of the source models. These self-supervised tasks are later used at test time, allowing

Table 1. Characteristics of various problem settings that deal with adapting a model (trained on a source distribution) to a test distribution. We propose SITA (Single Image Test-time Adaptation), which is the hardest adaptation setting.

Setting	Source Data	Target Label (y^t)	Train Loss	Test Loss	Offline	Online Statistics	Batched
fine-tuning	-	✓	$\mathcal{L}(x^t, y^t)$	-	✓	-	✓
domain adaptation	x^s, y^s	✗	$\mathcal{L}(x^s, y^s) + \mathcal{L}(x^t, x^s)$	-	✓	-	✓
SFDA	✗	✗	$\mathcal{L}(x^t)$	-	✓	-	✓
TTT [28]	x^s, y^s	✗	$\mathcal{L}(x^s, y^s) + \mathcal{L}(x^s)$	$\mathcal{L}(x^t)$	✗	✓	✓
FTTA [29]	✗	✗	✗	$\mathcal{L}(x^t)$	✗	✓	✓
SITA	✗	✗	✗	✗	✗	✗	✗

adaptation of the shared encoder between the main prediction branch and the auxiliary self-supervised task branch. TTT [28], one of the first works to formalize test time training, uses the self-supervised task of rotation prediction. Other recent works [1] extend the auxiliary branch based approach by using meta-learning to ensure that test-time training on the auxiliary branch would improve predictions from the main branch. These approaches cannot utilise any off-the-shelf model and improve its performance at test time. Instead they need to re-train the source model with handcrafted and often complicated training strategies, and are quite slow as they need back-propagation steps.

Fully Test Time Adaptation. These approaches have the advantage of adapting off-the-shelf models, since they do not rely on any auxiliary task. The state-of-the-art method in this category, TENT [29], formalised this setting and achieved better performance than some of the methods which actually re-train the source model. TENT runs an online optimization over a given test set, which minimizes the entropy of the predicted distribution for every incoming batch. Although the method can be made to work in the SITA setting where the model is reset after every test instance, their results are primarily focused in the online setting by gradually adapting the model over the test set. It gives impressive performance in the online setting, but we observe significant drop in performance in the SITA setting (Section 4), when there is only a single test instance. [18] improves upon TENT by adjusting the loss function to a log-likelihood ratio instead of entropy, and maintaining a running estimate of the output distribution. This setting, while clearly more useful than the previous one, is not as realistic as SITA because it needs to (i) accumulate and store samples to create batches, (ii) run an online optimisation under the assumption that the test samples come from the same distribution.

BatchNorm Adaptation. Recent literature on domain-adaptation shows that normalization parameters can have a significant impact on the performance in domain adaptation. Prediction Time Normalization (PTN) [19] and BN [26] builds on this idea and uses the mean and variance of the test instances for adaptation. This works reasonably well if the test batch size is large enough to provide a good estimate, but the performance drops significantly when we only have a single image. Moreover these works does not take into account the fact that different test instances might need different levels of adaptation, as they may be at different distances from the source distribution. Contrary to these works, our method first obtains a robust estimate of the batch-norm statistics from just a single image using label preserving transformations. Thereafter, it automatically identifies the optimal parameter, depending on the image, to combine the single image estimate with the source statistics, thus offering different levels of adaptation to different images. This makes our method applicable to the realistic and challenging SITA setting, showing consistent performance gains across a variety of datasets and tasks.

There are a few concurrent works related to TTA which recently appeared online. You *et al*. [32] use BN [26] with CORE [12] loss to adapt the affine parameters of the batch norm layer. Their method needs backpropagation and large batch sizes, not conforming with SITA. Zhang *et al*. [33] use 32/64 augmented samples for each test sample to arrive at a marginal output distribution and optimizes it using entropy loss similar to TENT [29], requiring an expensive optimization per sample. Hu *et al*. [10] maintain an online estimate for the statistics of the incoming test data along with augmentations. In comparison, our contributions are unique as we propose a lightweight adaptation technique, in accordance with the harder SITA setting. Further, most of these approaches involve hyper-parameter tuning for each target dataset, and the papers do not address tuning them at test-time, thus making them unrealistic in the SITA setting.

3. Methodology

We next discuss in detail the SITA setting and propose a simple, yet effective solution for the same. We first formally define the problem statement, and then discuss the challenges that motivate our approach.

Problem Statement: Consider that we have a model $f_\theta : \mathbf{x} \rightarrow \mathbf{y}$, trained on a dataset, which we refer to as the *source*. This model may not perform well on test data drawn from distributions other than the source, which can be either a corrupted version of the source itself or images drawn from a different distribution, both of which will be referred to as the *target*. The goal is to adapt the model f_θ such that the adapted model performs better on the target images compared to directly using the source model. As motivated in

Section 1, we focus on the challenging SITA setting, where inference has to be done on a single image at a time, and not a batch of instances, with the model being reset to the source model after every test instance adaptation.

Internal Covariate Shift: In deep neural networks, the mean and covariance statistics of every batch-normalization layer are learned using exponential moving average during training, and these accumulated statistics are usually used during testing, as given below for a particular layer:

$$\mathrm{BN}(\mathrm{F}) = \gamma \frac{\mathrm{F} - \mathbb{E}_s[\mathrm{F}]}{\sqrt{\mathrm{Var}_s[\mathrm{F}]}} + \beta \tag{1}$$

where F is the input feature to the batch norm layer and γ, β are scale and shift parameters, also learned during training. The subscript s indicates source data. The underlying reason behind using the train statistics instead of the test batch statistics is that the train statistics are estimated on a much larger set compared to a batch of test data, which is much smaller and hence can give a biased estimate of the statistics. While this method works well for the test samples drawn from a distribution similar to the source, its performance may degrade if the test samples come from a different distribution. This is because the feature distribution F of the intermediate layers of the network may have mean and variance shifted from the train statistics, due to the shift in the input distribution. This is often termed as internal covariate shift [11, 26].

To mitigate this problem, the statistics can be estimated from the target dataset and used, instead of the learned source statistics, i.e., replace $\mathbb{E}_s[\mathrm{F}]$ and $\mathrm{Var}_s[\mathrm{F}]$ in Eqn. (1) with $\mathbb{E}_t[\mathrm{F}]$ and $\mathrm{Var}_t[\mathrm{F}]$, which can be computed as follows:

$$\begin{aligned} \mathbb{E}_t[\mathrm{F}] &= \frac{1}{n_t HW} \sum_{i,j,k} \mathrm{F}_{i,j,k} \\ \mathrm{Var}_t[\mathrm{F}] &= \frac{1}{n_t HW} \sum_{i,j,k} (\mathrm{F}_{i,j,k} - \mathbb{E}_t[\mathrm{F}])^2 \end{aligned} \tag{2}$$

$\mathrm{F}_i \in \mathbb{R}^{H,W,F}$ is the feature map of the i^{th} instance. H, W, F are the height, width and feature dimension respectively. $j \in \{1, \ldots, W\}, k \in \{1, \ldots, H\}$ and n_t is the number of target instances in the dataset. This is strategy is followed in several domain adaptation works [2, 14], as well as in Prediction Time Normalization (PTN) [19]. The domain adaptation works assume access to a target training set with which one can obtain a good estimate of the above mean and variance. On the other hand PTN assumes a huge batch of test instances, thus able to obtain a good estimate of the mean and variance. However, in the SITA setting, neither do we have access to a target training set or a batch of test instances, and cannot use the updated model or the current instance's statistics to infer the subsequent test instances. This makes our setting much harder compared to other settings in literature on test time adaptation [29].

Batch-Norm Parameter Calibration: When we have a limited number of target samples from a distribution different from the source, using only the target statistics from Eqn 2 may not work well, because of unreliable estimates from only a few samples. Additionally, using only the source statistics may not work well because of internal covariate shift. Instead using a weighted combination of the two may be a better alternative:

$$\begin{aligned} \mu &= \lambda \mathbb{E}_s[\mathrm{F}] + (1-\lambda)\mu_t \\ \sigma^2 &= \lambda \mathrm{Var}_s[\mathrm{F}] + (1-\lambda)\sigma_t^2 \end{aligned} \tag{3}$$

where $\lambda \in [0, 1]$. This strategy considers the source statistics as a prior. On one hand, $\lambda = 1$ results in using the source model directly on the target instances. In this case, the estimator has high bias. On the other hand, $\lambda = 0$ results in using only the single test image statistics. In this case, the variance of the estimator is high. The estimator in Eqn. (3) provides a balance to bias and variance, with the underlying hypothesis that the target distribution is a shifted version of the source. It can be shown that the mean estimator has $(1-\lambda)^2$ times lower variance and $||(1-\lambda)(\mathbb{E}_s[\mathrm{F}] - \mathbb{E}_t[\mathrm{F}])||$ lower bias magnitude than the worst variance and bias of the two estimators corresponding to $\lambda = 1$ and $\lambda = 0$.

While this may perform well when the is at least a few target instances, in the SITA setting, with only one test instance, the estimation error can be significant.

Augmentation for Statistics Estimation: Given a single test instance, we can estimate its statistics (μ_t, σ_t) using Eqn. (2), with $n_t = 1$. However, this estimate may have errors as it is computed using a single instance. Ideally, we want the single image estimate (μ_t, σ_t) to be as close as possible to the true statistics of the target distribution. The variance of the above estimators can be reduced by increasing the number of samples. Given that we have only one sample $\mathbf{x}$ at hand, we ask the question - *is it possible to generate more data points to improve the estimates?* While we do not have access to the underlying distribution of the target instances, we can possibly create data points in the vicinity of the test sample $\mathbf{x}$. Inspired by the recent works in contrastive learning for representation learning [4] and theoretical justifications of pseudo-labeling [30], which suggests that using neighborhood samples via augmentation helps in the respective tasks, we adopt this idea for our task to improve the estimate of target image statistics.

Specifically, we use a set of augmentations to augment $\mathbf{x}$ and obtain $\{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n\}$. This generates features $\{\hat{\mathrm{F}}_1, \ldots, \hat{\mathrm{F}}_n\}$ for a certain batch normalization layer, and we use these features along with the original feature F to obtain a better estimate of the statistics, μ_t and σ_t from the given single image. However, as it is hard to control the distribution of the augmented samples, and certain augmentations can outweigh the estimation, instead of assigning the

same weight to all the augmented samples as the original sample, we distribute the weight as follows:

$$\mu_t = \mathbb{E}_w(\{\mathrm{F}, \hat{\mathrm{F}}_1, \ldots, \hat{\mathrm{F}}_n\}; w = \{1/2, 1/2n, \ldots, 1/2n\})$$
$$\sigma_t = \mathrm{Var}_w(\{\mathrm{F}, \hat{\mathrm{F}}_1, \ldots, \hat{\mathrm{F}}_n\}; w = \{1/2, 1/2n, \ldots, 1/2n\}) \quad (4)$$

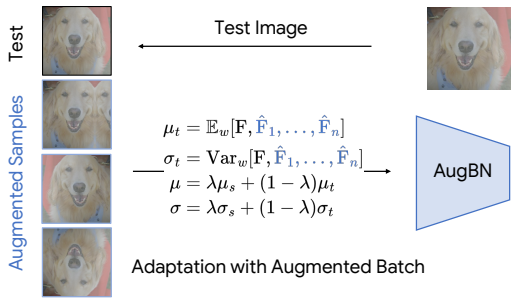


Figure 2. **AugBN layer.** Modifying the BatchNorm (BN) layer with augmented samples (Algorithm 2). The test image along with a combination of augmented samples are used to estimate the batch-norm statistics using Eqn. 4. This estimation happens with one forward pass using the AugBN layer (Algorithm 1).

We choose a fixed set of augmentations $\{a_1, \ldots, a_m\}$, from which we randomly pick $k(\leq m)$ augmentations. These are composed to obtain the augmentation functions A_i, and apply it on the test image to generate augmented images $\hat{\mathbf{x}}_i$. Since these new samples $\hat{\mathbf{x}}_i$ are generated from a single parent sample $\hat{\mathbf{x}}$, the samples are not independent. Thus, the variance reduction may not be linear with the number of augmented samples (as the underlying assumption in that case is that the samples are independent). Having said that, using the augmented samples does improve the performance over a range of tasks, as we observe in our experiments (Section 4). The modifications needed in the vanilla BatchNorm layer to incorporate the AugBN layer is described in Algorithm 1. Figure 2 shows AugBN pictorially. In the next section, we propose how to set the prior parameter λ, optimally for each test instance using a single forward pass.

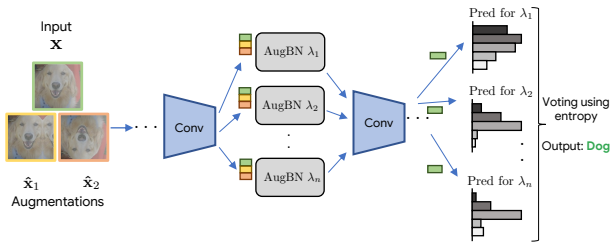


Figure 3. Illustration for the SITA adaptation method. Each AugBN layer maintains a vector of statistics for each prior choice. The model returns the final prediction based on a majority vote on the entropy of the output distribution (OPS).

Optimal Prior Selection (OPS): Test instances can originate from different distributions, which may be at different distances from the source. We ask this question - *"Can we automatically identify the level of adaptation needed, that too in a single forward pass?"*. To answer this question, we look at the prior parameter λ in Eqn 3, which combines the source and the target statistics. Optimally choosing this parameter conditioned on the test image can allow us to efficiently control the level of adaptation.

Algorithm 1: AugBN

Input: - Source statistics: μ_s, σ_s, prior $= \lambda$
- Input batch features: F, $\oplus\ \{\hat{\mathrm{F}}\}_{i=1}^n$.
Output: Normalised Features:
$\bar{\mathrm{F}} = \mathrm{AugBN}(\mathrm{F}, \{\hat{\mathrm{F}}_i\}_{i=1}^n)$
$\ominus\ \bar{\mathrm{F}} \leftarrow \frac{\mathrm{F}-\mu_s}{\sigma_s}$
$\oplus\ \mu \leftarrow \lambda\mu_s + (1-\lambda)\mu_t$ (Eqn. 4)
$\oplus\ \sigma \leftarrow \lambda\sigma_s + (1-\lambda)\sigma_t$ (Eqn. 4)
$\oplus\ \bar{\mathrm{F}} \leftarrow \frac{\mathrm{F}-\mu}{\sigma}$

Comment: $\oplus$ and $\ominus$ signify additions and removals from the standard batch normalization layer)

Algorithm 2: Proposed Algorithm for SITA

Input: - Single Image: $\mathbf{x}$
- Source Model: f_θ
Output: Prediction: $\mathbf{y}$
$\hat{f}_\theta \leftarrow$ Repalce BN Layer in f_θ with AugBN Layer
for $i = 1 \ldots n$ **do**
  $\mathrm{A}_i \leftarrow$ Compose(RAND-Choose-k($\{a_i\}_{i=1}^m$))
  $\hat{\mathbf{x}}_i \leftarrow \mathrm{A}_i(\mathbf{x})$
end
$\mathbf{X} \leftarrow$ REPEAT$[\mathbf{x}, \hat{\mathbf{x}}_{1:n}]$ k times
$\boldsymbol{z} \leftarrow \hat{f}_\theta(\mathbf{X})$ s.t. λ_i used for $\mathbf{X}_{i(n+1):(i+1)(n+1)}$
$\boldsymbol{y} \leftarrow \mathrm{OPS}(\boldsymbol{z})$

1. RAND-Choose-k() uniformly randomly chooses $k(< N)$ augmentations from the given set.
2. OPS() denote the Optimal Prior Selection strategy.

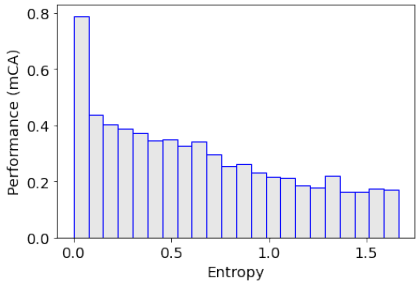


Figure 4. Variation of model performance with entropy. We observe a strong negative correlation between entropy of an instance's prediction and its correctness, and thus use entropy as a metric to determine the level of adaptation needed.

With this goal in mind, we envisage the entropy of the prediction's probability mass function as a metric to measure its correctness. In Fig. 4, we observe a strong negative correlation between correctness of prediction and entropy. Specifically, for classification, we forward propagate through an AugBN equipped network for n_p different prior values (batched together in a single forward pass) and first choose the k ($k < n_p$) priors which have the lowest entropy of the predictions. We then take a majority voting of the predictions for these k priors, and break a tie with the lowest entropy value. For segmentation, we repeat this strategy for all pixels individually. Note that we repeat the same set of augmented images for all the n_p priors. For all datasets/tasks, we use $n_p = 8$, and $k = 3$. This approach not only allows us to choose the level of adaptation based on the input, but also automatically chooses the prior λ, which would otherwise be an hyperparameter. Figure 3 illustrates the single forward pass for our approach, which makes it practical in the SITA setting. The entire algorithm to adapt the model during prediction is shown in Algorithm 2.

4. Experiments

Here we demonstrate the effectiveness of our approach by running thorough experimentation on a variety of adaptation benchmarks for both segmentation and classification, and over multiple network architectures. In comparison, the focus in most related works has primarily been applied on classification tasks, with limited number of networks.

Semantic Segmentation Datasets: We evaluate our method on three different source $\rightarrow$ target combinations covering both indoor and outdoor scenes, to showcase the wide usability of our algorithm. For outdoor, we evaluate on GTA5 [22] $\rightarrow$ Cityscapes [5] and SYNTHIA [23] $\rightarrow$ Cityscapes. For indoor, we show results on SceneNet [17] $\rightarrow$ SUN [27]. The outdoor and indoor scene datasets have 19 and 13 categories, respectively. Following the literature on domain adaptation of semantic segmentation models, we use Deeplab-V2 [3] with ResNet-101 [6] as the backbone. We use one GPU to train source models with a batch size of 1 in all experiments. We use SGD with an initial learning rate of 2.5×10^{-4} with polynomial decay of power 0.9 [3]. We use the standard metric of mean intersection over union (mIoU) [3]. We use Gaussian filtering and random rotation as augmentations as they preserve the are label preserving for segmentation.

Classification Datasets: Following the literature [18, 26, 28, 29], we evaluate our method on common adaptation datasets, namely the corruption and perturbation data, CIFAR-10-C and ImageNet-C [8]. Following previous works, we report results on the highest severity level of corruption. We compare the mean Classification Accuracy (mCA) over all 15 corruption types on both the datasets. Further, we test our approach on ImageNet-R and ImageNet-A, comprising of renditions of ImageNet classes and adversarial ImageNet examples respectively. We adopt the same network architecture used in the recent works [28, 29], i.e., ResNet-26 and ResNet-50 for CIFAR-10 and ImageNet respectively. The source model for CIFAR-10 is trained on one GPU with a batch size of 64, using SGD with cosine decay scheduler and an initial learning rate of 0.01. The source model achieves 93.84% accuracy on the CIFAR-10 test set. For ImageNet, the model is trained on 8x8 TPU slices with a batch size of 8192. We use Adam optimizer along with the cosine decay scheduler with an initial learning rate of 0.1. This source model obtains 76.4% accuracy on ImageNet. For all classification results we use two augmented samples, each composed of five SimCLR [4] based augmentations from color distortion [28], rotation, mirror reflection, vertical and horizontal flip augmentations which are randomly shuffled, creating a diverse set of label preserving augmentations which are used for all datasets and network architectures without any tuning. Please refer to the supplementary for more details.

Baselines. We compare with the strong state-of-the-art baselines, namely TENT [29], BN [26], and Prediction Time Normalization (PTN) [19], and Aug-Ensemble. In Aug-Ensemble, we follow a common practice and augment the test instance to obtain multiple augmented images, and then take an average over all the predictions to get the final prediction. For classification we use the same augments in AugEnsemble as we use in our algorithm. But for segmentation, to maintain the spatial correspondence, we only use gaussian smoothing and gaussian noise as the augmentations for AugEnsemble. All methods are executed in the SITA setting, that is, the test batch size for all experiments is fixed to 1 and no method is allowed to run an online optimization or to maintain online statistics. Note that the results in the TENT paper [29] are for the online setting over a batch size of 64 and 128 for ImageNet-C and CIFAR-10-C respectively. But we apply TENT in the SITA setting, where we reset the model to the source model after inferring every test instance, and use five iterations of optimization with a learning rate of 10^{-3} (this is the best result we obtained by experimenting over a range between $[10^{-2}, 10^{-5}]$) with Adam optimizer as it provides the best computation time vs performance trade-off. For BN, we showcase the results with the suggested hyper-parameter setting ($N = 16$ in [26]).

Layer-wise analysis. To study the effect of adapting different layers of a network and which layers are the most affected by the adaptation, we design an experiment on CIFAR-10-C using ResNet-26, which has 4 block-groups, grouped by number of features (64, 128, 256, 512), each consisting of 3 residual blocks. We replace source BN layers with AugBN layers for each residual block and observe the performance (Figure 5). The results for all configurations are shown in

Table 2. Comparison of our method (SITA) with other baselines for test time adaptation with a single image at a time.

Methods	Segmentation (mIoU)			Classification (mCA)			
	GTA5 → Cityscapes	SYNTHIA → Cityscapes	SceneNet → SUN	CIFAR10-C	ImageNet-C	ImageNet-A	ImageNet-R
Source Model	37.6	32.1	26.5	61.4	20.5	0.5	35.4
Aug Ensemble	35.5	32.4	26.7	61.3	20.5	0.6	35.2
PTN [19, 26]	36.7	31.8	25.2	55.8	0.3	0	1.4
BN [26]	40.4	32.9	28.2	65.1	24.5	0.9	38.4
TENT [29]	37.6	32.7	26.6	55.9	0.3	0	1.2
SITA	**42.9**	**36.5**	**28.8**	**73.1**	**25.5**	**1.1**	**40.3**

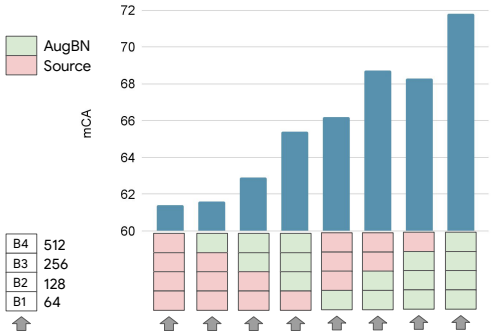


Figure 5. Results for layer-wise sensitivity analysis.

Figure 5. First column on the left represents all blocks using source BN layers, and the last column represents using AugBN in all the layers. In columns 2-4, we set the final blocks to gradually use AugBN and in columns 5-7 we set the initial blocks to use AugBN. As columns 5-7 has much better performance than columns 2-4, it shows that adapting the first few layers has more impact than adapting only the last few layers, as the former capture domain specific information. Moreover, using AugBN in only the first block (column 5), we obtain higher mCA than using the source statistics in the first block and AugBN on the rest.

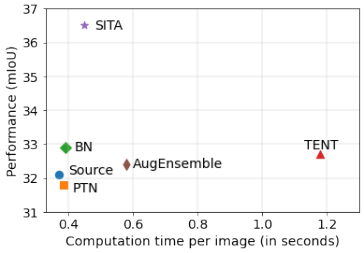
Comparison with state-of-the-art. Table 2 shows the comparison with state-of-the-art methods for both classification and segmentation. For segmentation, our method SITA achieves the best performance on all the three datasets with a 14.8%, 14.0% and 8.8% relative improvement over the source model on GTA5 → Cityscapes, and SYNTHIA → Cityscapes and → SUN respectively. For classification, our method gives a huge 19.1% relative improvement on CIFAR-10-C compared to just using the source model directly, and also performs much better than state-of-the-art approaches. ImageNet-C is a much harder dataset to do adaptation using just a single instance, and the methods which do not incorporate source statistics (TENT [29] and PTN [19]) suffer a huge loss in performance as their normalisation statistics are highly incompatible with the source model.

Ablation of Optimal Prior Selection. While a network equipped with our AugBN layer may perform good for a range of prior values, the best performance may not be at the same fixed prior value across all datasets, thus making it tricky to choose the best prior without the presence of a validation set. For e.g., Figure 7 shows that the best performing prior is $\lambda = 0.5$ for SYNTHIA → Cityscapes, but $\lambda = 0.6$ on CIFAR-10 → CIFAR-10-C. This is because different target datasets, and even different target samples, maybe at different distances from the source distribution, thus requiring different levels of mixing between the source and target statistics. However, the optimal prior selection strategy in our algorithm automatically selects the prior for every sample, and perform at par or even better than using the best fixed prior on CIFAR-10-C.

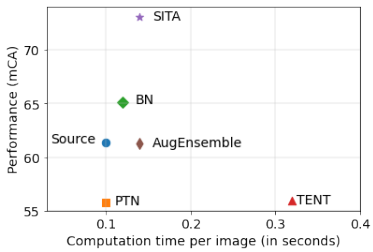
Computational Analysis The proposed method requires only one forward pass of the original test image itself along with its augmented versions, and thus the inference time is very close to the source model itself. We report the average computation time for all methods on the SYNTHIA [23] → Cityscapes setting in Figure 6a and on CIFAR-10-C in Figure 6b. This computation time also includes the time required to compute the necessary augmentations. Compared to TENT [29] which is currently the state-of-the-art TTA method, our method is 2.2x faster with a 11.6% relative improvement on segmentation and (Table 2) and 2.5x faster with a 30.7% improvement in classification.

Results on additional architectures: To showcase the universality of our approach we test our adaptation technique across various pretrained ImageNet network architectures on the ImageNet-C, ImageNet-A and ImageNet-R in Table 3. Our approach provides significant performance gains across all architectures and datasets. This shows the generalizability of our method across architectures.

Generalized Test Time Adaptation: In practice, the test instances can originate from both the source as well as unknown target distributions. Ideally TTA algorithms should

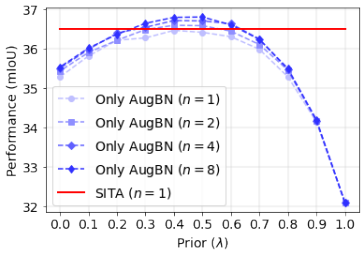


(a) SYNTHIA → Cityscapes

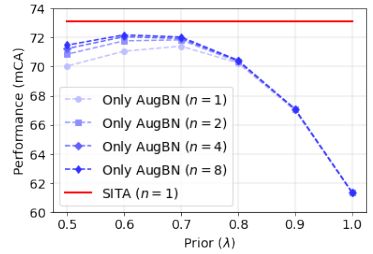


(b) CIFAR-10 → CIFAR10-C

Figure 6. Computation time vs performance comparison on (a) segmentation and (b) classification tasks. Our method take almost similar time as the source model's inference time, but performs much better than all the baselines.



(a) SYNTHIA → Cityscapes.



(b) CIFAR10 → CIFAR10-C dataset.

Figure 7. Ablation of automatically selecting the prior parameter for every image (red) vs using a fixed parameter for all images (blue) denoted by "Only AugBN". The numbers in brackets are the number of augmented samples used. The performance varies significantly with the choice of fixed prior value, and it becomes difficult to choose the best fixed prior parameter without a validation set. Whereas, our method automatically selects the prior for every image, and can perform even better than the best fixed parameter, as it has the ability to judge the level of adaptation needed depending on the input sample.

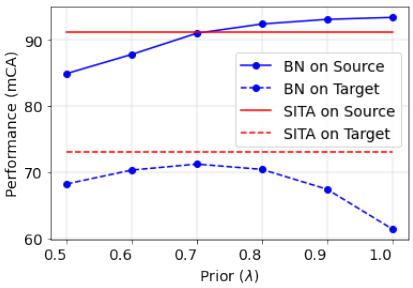


Figure 8. **Generalized TTA**: The λ parameter in BN [26] that works well for images from source distribution (CIFAR10) is different from the one that works well for the target distribution (CIFAR-10-C), which is a sticky issue. On the other hand, the proposed method SITA (horizontal red lines) does not need a curated λ value, instead it automatically chooses the best λ for each test instance.

work well on both the cases. In the SITA setting, we have no information about the originating distribution, and thus the BN method [26], which calibrates the batch-norm statistics using a fixed prior, may not work well if a test instance from the source appears (Figure 8). This is because the source statistics would work the best in such a case, rather than the calibrated statistics. However, the proposed OPS strategy would automatically choose a suitable prior parameter λ irrespective of whether the test instance is from unknown target distribution or the source distribution. Hence the proposed approach is more robust in such realistic situations.

Table 3. Results on other network architectures. -C, -A, and -R denote ImageNet-C, ImageNet-A and ImageNet-R datasets.

Networks →	DenseNet121			InceptionV3		
Datasets →	- C	- A	- R	- C	- A	- R
Source	22.7	0.7	37.5	30.4	3.2	39.1
PTN [19, 26]	0.1	0.0	0.6	0.2	0.1	1.0
BN [26]	25.6	1.3	40.2	35.8	3.5	42.8
SITA	**25.8**	**1.4**	**41.4**	**36.2**	**3.7**	**46.0**

5. Conclusion

Test-time adaptation has gained recent interest in the literature given its ability to adapt models only during test-time. In this work, we formalise the test time problem under the realistic and challenging **S**ingle **I**mage **T**est-time **A**daptation (SITA) setting. We propose a simple framework which significantly outperforms the state-of-the-art while overcoming some of the limitations of the existing approaches. The strength of the proposed approach is in its speed and performance, applicability for multiple tasks and network architectures, while being simple and easy to implement.

References

[1] Alexander Bartler, Andreas Bühler, Felix Wiewel, Mario Döbler, and Binh Yang. Mt3: Meta test-time training for self-supervised test-time adaption. *CoRR*, abs/2103.16201, 2021. 3

[2] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *CVPR*, 2019. 4

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017. 6

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 4, 6

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 6

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

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

[8] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019. 2, 6

[9] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020. 1

[10] Xuefeng Hu, M. Gökhan Uzunbas, Sirius Chen, Rui Wang, Ashish Shah, Ram Nevatia, and Ser-Nam Lim. Mixnorm: Test-time adaptation through online normalization estimation. *CoRR*, abs/2110.11478, 2021. 3

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[12] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. In *ECCV*, 2020. 3

[13] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *CVPR*, 2020. 2

[14] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *ICLRW*, 2017. 4

[15] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020. 2

[16] Yuang Liu, Wei Zhang, and Jun Wang. Source-free domain adaptation for semantic segmentation. In *CVPR*, 2021. 2

[17] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *ICCV*, 2017. 6

[18] Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*, 2021. 3, 6

[19] Zachary Nado, Shreyas Padhy, D. Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *CoRR*, 2020. 3, 4, 6, 7, 8

[20] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *CVPR*, pages 12556–12565, 2020. 1

[21] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, pages 5389–5400, 2019. 1

[22] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 6

[23] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 6, 7

[24] Prabhu Teja S and Francois Fleuret. Uncertainty reduction for model adaptation in semantic segmentation. In *CVPR*, 2021. 2

[25] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *ICLR*, 2020. 1

[26] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. 2, 3, 4, 6, 7, 8

[27] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 6

[28] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020. 1, 2, 3, 6

[29] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 1, 2, 3, 4, 6, 7

[30] Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. *ICLR*, 2021. 4

[31] Hao-Wei Yeh, Baoyao Yang, Pong C Yuen, and Tatsuya Harada. Sofa: Source-data-free feature alignment for unsupervised domain adaptation. In *WACV*, 2021. 2

[32] Fuming You, Jingjing Li, and Zhou Zhao. Test-time batch statistics calibration for covariate shift. *CoRR*, abs/2110.04065, 2021. 3

[33] Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: test time robustness via adaptation and augmentation. *CoRR*, abs/2110.09506, 2021. 3