Bag of Tricks for Fully Test-Time Adaptation

Saypraseuth Mounsaveng[*], Florent Chiaroni, Malik Boudiaf, Marco Pedersoli, Ismail Ben Ayed
ÉTS Montréal, Canada

Abstract

Fully Test-Time Adaptation (TTA), which aims at adapting models to data drifts, has recently attracted wide interest. Numerous tricks and techniques have been proposed to ensure robust learning on arbitrary streams of unlabeled data. However, assessing the true impact of each individual technique and obtaining a fair comparison still constitutes a significant challenge. To help consolidate the community's knowledge, we present a categorization of selected orthogonal TTA techniques, including small batch normalization, stream rebalancing, reliable sample selection, and network confidence calibration. We meticulously dissect the effect of each approach on different scenarios of interest. Through our analysis, we shed light on trade-offs induced by those techniques between accuracy, the computational power required, and model complexity. We also uncover the synergy that arises when combining techniques and are able to establish new state-of-the-art results.

1. Introduction

Deep neural networks perform well at inference time when test data comes from the same distribution as training data. However, they become inaccurate when there is a distribution shift [25]. This distribution shift can be caused by natural variations [14] or corruptions [9, 10]. Test-Time Adaptation (TTA) aims at addressing this problem by adapting a model pre-trained on source data to make better predictions on shifted target data [2, 13, 28]. In this work, we focus on the particular case of Fully Test-Time Adaptation (Fully TTA) [22, 30, 36]. In this setting, the adaptation is done source free and relies only on: i) a model pre-trained on data from a source domain and ii) unlabeled test data from a shifted target domain. Separating the training phase from the adaptation phase is particularly relevant for privacy-oriented applications where the training data is not available or can not be disclosed. Fully TTA is also online. Test data is received as a continuous stream and the model adaptation is done on-the-fly as data is received. This makes the setup more realistic and closer to real-world ”in-the-wild” scenarios where information about potential distribution shifts or about the quantity of data to be received is not necessarily available.

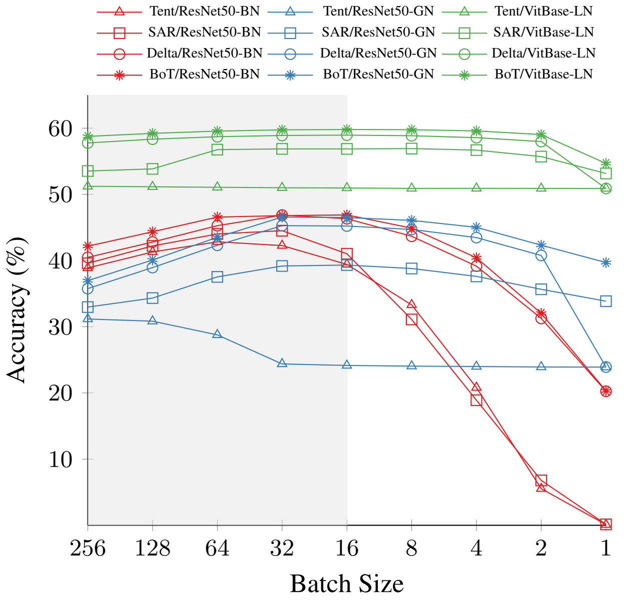


Figure 1. **Classification Accuracy in function of Batch Size for different methods and architectures on ImageNet-C.** In this work, we choose to focus on small batches (16 and below, white zone). As the batch size decreases, the model performances remain stable until a batch size of 32 and then drops significantly for methods running on ResNet50-BN. Results reported are averaged over 15 corruptions and 3 runs. Confidence intervals are too small to be displayed.

Most of the recent solutions proposed to address Fully TTA are follow-ups of seminal work Tent [30] and aim at solving problems inherent to the online and unsupervised aspect of Fully TTA. For example, [32, 36] deal with the problem of the class imbalance data stream, [22, 35] improve the quality of the predictions used to adapt a model by selecting samples with a low entropy or leveraging the predictions of augmented samples and [18, 35, 35, 36] investigate different normalization to stabilize the adaptation

[*] Corresponding author: saypraseuth.mounsaveng.1@etsmtl.net

[†] Code is available at https://github.com/smounsav/tta_bot

process. However, most of the tricks and techniques are presented in combination with others, which makes it difficult to identify their impact on the final model performance. Some techniques might already help when applied alone whereas others might only work or work better in combination with other tricks. As this area of research is very active and developing fast, we aim in this study at disentangling the impact of some techniques recently proposed and evaluate objectively their contribution to the performance of Fully TTA models. We also propose possible improvements in specific cases.

Contribution. To address the Fully Test-Time Adaptation problem, we analyzed the following techniques: i) Usage of batch renormalization or batch-agnostic normalization ii) Class re-balancing iii) Entropy-based sample selection iv) Temperature scaling. Those analyses were made considering small batch sizes (16 and below), which are closer to the potentially uncontrollable batch sizes of real-world scenarios. Our experimental results show that those techniques are already boosting the performance at test time when used alone, but that combining all of them leads to the best classification accuracy compared to a vanilla Tent method and 2 recent state-of-the-art methods on 4 different datasets. Additionally, to the accuracy improvement, the selected techniques also bring other interesting benefits like higher and more stable performance with small batch sizes and a reduced computational load by adapting the model with a reduced set of selected data.

The remainder of the paper is structured as follows. We conduct a literature review in Section 2. Then we analyze each trick separately in a different section: architecture design in Sec. 4, class rebalancing in Sec 5, sample selection in Sec. 6 and network calibration in Sec. 7 before showing results on combinations of tricks in Sec. 8 and results on other datasets in Sec. 9. Finally, we conclude about the presented work in Sec. 10.

2. Related Work

Test-time adaptation (TTA). Test-time adaptation assumes access to a pre-trained model and aims at leveraging unlabeled test instances from a (shifted) target distribution to make better predictions. Proposed methods usually employ one or a combination of the following techniques: *self-training* to reinforce the model's own predictions through entropy minimization [30] or Pseudo-Labelling schemes [15], *manifold regularization* to enforce smoother decision boundaries through data augmentation [35] or clustering [4], *feature alignment* to mitigate covariate shift by batch norm statistic adaptation [16, 27], and *meta-learning* methods [6] that try to meta-learn the best adaptation loss.

TTA in the broader literature. Although recently introduced [30], TTA shares important motivations and similarities with earlier or concurrent settings that are source-free domain adaptation (SFDA) [3, 17, 34] and test-time training (TTT) [23, 28]. In SFDA, methods also leverage samples from the target distribution of interest but have no access to source data, and the evaluation is still done on held-out test data. In other words, TTA is the transductive counterpart of SFDA. On the other hand, TTT works by constructing an auxiliary task that can be solved both at training and adaptation time and therefore, unlike TTA, is not agnostic to the training procedure or to the model architecture.

Fully TTA. TTA is of particular interest for online applications, in which the model receives samples as a stream. Operational requirements for online applications break crucial properties of the vanilla TTA setting e.g. large batch size or class balance. Under such operational requirements, standard TTA methods degrade, underperforming the non-adapted baseline and even degenerating to random performance in some cases [4, 22]. Multiple regularization procedures have been proposed to address such shortcomings. Among them, (i) Improved feature alignment procedures that interpolate, between source and target statistics [18, 20, 36], thereby improving overall estimation and decreasing reliance upon specific test batches, (ii) Sample reweighting [21, 36] to alleviate the influence of class biases, (iii) Improving loss' intrinsic robustness to noisy samples, either encouraging convergence towards local minima [22] or preventing large deviations from the base model's predictions [4, 21]. Recently, [29] explored the update of the model weights using Hebbian learning instead of just updating the BatchNorm layers. As this line of work grows, the current study provides an objective evaluation of how recently proposed ingredients translate into actual robustness for Fully TTA and quantifies the progress made so far, as well as pinpoints possible areas of improvement. A detailed comparison of the Fully TTA setting with the other TTA settings is available in the supplementary material.

3. Experimental Setup

In this section, we present the details of our experimental setup. Firstly, we introduce the datasets used, then the different methods we want to compare and the different models, and finally, we explain the evaluation metric and protocol. For reproducibility purposes, the links to the code and model weights used in our experiments are provided in the supplementary material.

3.1. Datasets

We evaluate the different methods on several datasets used by prior SFDA or TTA studies: (i) ImageNet-C [10] is a variant of ImageNet [26] where 19 corruption types and 5 levels of severity were applied. For our experiments, we report results using 15 corruption types at the most severe level of corruption (level 5) and keep the 4 remaining extra (speckle noise, gaussian blur, spatter, and saturate) as ”validation” corruptions to select hyperparameters following [36] and [22]. (ii) ImageNet-Rendition [9] consists of 30,000 images distributed in 200 Imagenet classes obtained by the rendition of ImageNet images like art, cartoons, tattoos, or video games. (iv) ImageNet-Sketch [31] is a dataset of 50,0000 images distributed in all ImageNet classes and obtained by querying Google Images with ”sketch of __” where __ is the name of original ImageNet classes. Images are in the black and white color scheme. (v) Finally, VisDA2017 [24] is a dataset of over 72K images distributed in 12 ImageNet classes and containing a mix of synthetic and real domain images. In the sections where we analyze tricks (Class rebalancing Sec. 5, Sample Selection Sec. 6, Calibration Sec. 7, and Tricks combination Sec. 8), all experiments are done using ImageNet-C.

3.2. Methods

In this work, we chose to analyze the following tricks and methods: (i) Tent [30] is a seminal work in Fully Test-Time Adaptation and is the first work to use an entropy-based loss in the adaptation process. (ii) SAR [22] is a state-of-the-art method in Fully TTA and proposes a method to select the most useful samples based on their entropy. (iii) Delta [36] is also a state-of-the-art method in Fully TTA and focuses on addressing the problem of online class rebalancing. (iv) in our experimental setup, we call BoT the model combining the best tricks selected in the different experiments.

3.3. Models

In our experiments, we use different architectures depending on the datasets tested. In experiments with ImageNet-C, we follow [22] and use two variants of the ResNet50 architecture [8] and a ViT-Base/16 transformer architecture. The first ResNet50 variant (ResNet50-BN) uses batch normalization layers [12] whereas the second one (ResNet50-GN) uses group normalization [33] layers. The ViTBase/16 transformer uses layer normalization [1] and will be referred to as VitBase-LN. For experiments with VisDA2017, we follow [34] and [3] and use a ResNet101 architecture. The number of parameters of each architecture is available in the supplementary material.

3.4. Evaluation metrics

To evaluate the different approaches, we use the classification accuracy metric. To compute this metric, we follow [22] and [36] and consider the accumulated predictions of the test samples after each model update. In other words, we do not compute the classification accuracy on the whole test set after the model has seen all test samples but online after each batch. Results reported are averaged over 3 runs.

4. Architecture and Normalization

In this section, we investigate the influence of different architectures and normalization on the model performance. Normalization in particular has been an active area of research in the TTA literature. [36] shows that in the case of a distribution shift, normalization statistics are inaccurate within test mini-batches and the gradient of the loss can show strong fluctuations potentially destructive for the model. To address this issue, [18] proposes to combine linearly the statistics learned during training with the statistics computed at test time to reduce the gap between the source domain and the target domain. However, this method is not applicable in Fully TTA as it requires access to labeled source data to learn the linear combination in a post-training phase before using it at test time. [19, 35] also use a linear combination of the training statistics and the test statistics to handle the distribution shift. [36] adapts batch renormalization [11] to test-time adaptation. Batch normalization parameters are updated using a combination of the mini-batch statistics and moving averages of these statistics like in the original paper, but in the TTA context, statistics and moving averages are computed using test batches. Another way to address the issues inherent to batch normalization is to use group or layer normalization instead as investigated in [22]. As the normalization differs a lot between works, this study aims at disentangling its effect from other techniques used.

In our experiments, we follow [22] and use the following architectures: i) a ResNet50 with BatchNorm layers (ResNet50-BN) ii) a ResNet50 with GroupNorm layers (ResNet50-GN) iii) a VitBase/16 with LayerNorm layers (VitBase-LN) iv) to complete our pool of models to compare, we also include a variant of ResNet50-BN where batch normalization is replaced by batch renormalization (ResNet50-BReN).

Experimental results In Fig. 2, we observe that the performance of Tent method on a ResNet50-BN architecture is dropping when the batch size is becoming small, with a particularly low performance when the batch size is 2 (5.53% accuracy) or 1 (0.14% accuracy). Intuitively, those results can be explained by the fact that batch normalization layers are normalizing the weights based on the statistics of the current batch. When the batch becomes too small, the statistics computed have a high variance, are not representative anymore of the test distribution and are not informative enough about the domain shift. However, we see that using batch renormalization instead of standard batch normalization improves the performance of a ResNet50 model and avoids a complete collapse of the model when the batch size is 1. Also in Fig. 2, we observe that Tent performance on

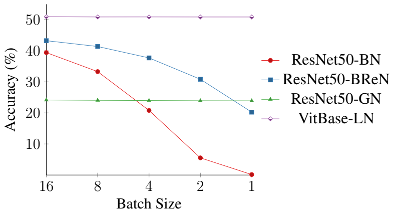


Figure 2. **Impact of Normalization, Architecture, and Batch Size on classification accuracy of Tent method on ImageNet-C.** Using a batch renormalization layer leads to better performance than using a vanilla batch normalization. Tent performance is more stable on architectures with batch-agnostic normalization like group or layer normalization.

architectures with batch-agnostic normalization layers such as GroupNorm or LayerNorm is more stable and less impacted by a reduction of the batch size.

5. Class rebalancing

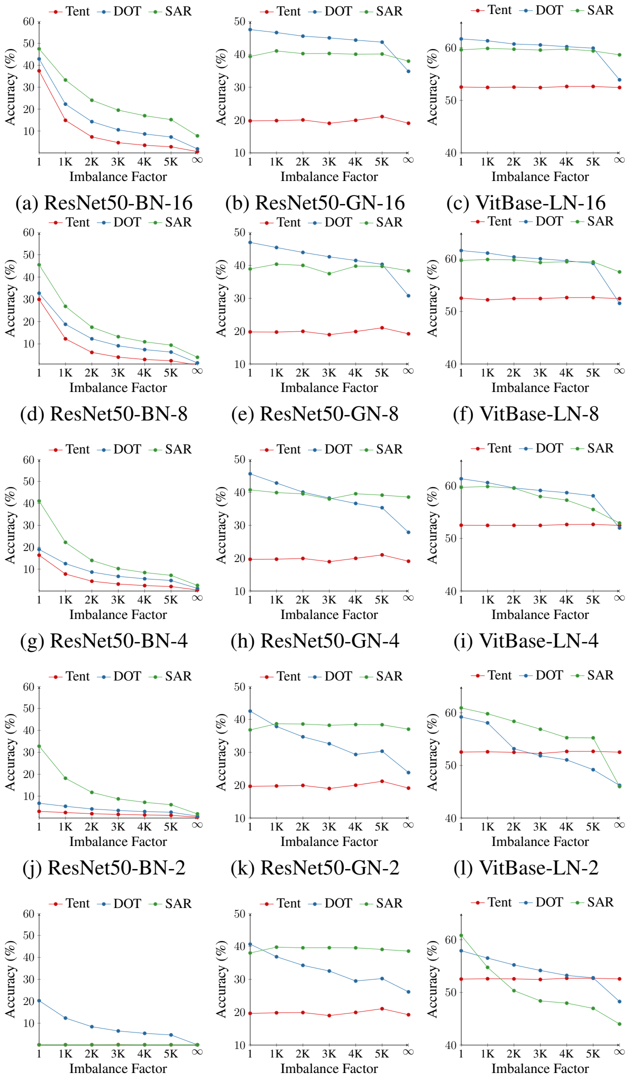
In this section, we explore the problem of online class imbalance in the context of Fully TTA. This problem is strongly relevant in this setting as data is received as a continuous stream. In this case, there is no guarantee that classes will appear in a balanced way or that different classes will appear in a given batch, especially when the batch size becomes much smaller than the total number of classes in the dataset. Imbalanced data can be particularly detrimental to the model performance as shown in [22, 32, 36] and can lead in extreme cases to a model collapse to trivial solutions like assigning all samples to the dominant class.

To evaluate methods in regard to this problem, we consider two approaches. In the first one, we follow the setup proposed in [22]. In this setup, the online imbalanced label distribution shift is simulated by controlling the order of the input samples using a dataset generated using the following sampling strategy: a probability vector $Q_t(y) = [q_1, q_2, ..., q_K]$ is defined, where t is a time step and T is the total number of steps and is equal to K the total number of classes, and $q_k = q_{max}$ if $k = t$ and $q_k = q_{min} \triangleq (1 - q_{max})/(K - 1)$ if $k \neq t$. The ratio q_{max}/q_{min} represents the imbalance ratio. For ImageNet-C, at each time step $t \in 1, 2, ..., T = K$, 100 images are sampled using $Q_t(y)$ and so in total, the dataset contains 100x1000 images. An imbalance factor of 500000 is represented in Fig. 3 as ∞ and represents a setup very close to the adaptation of the model one class after the other. Then, in a second approach, we investigate the evolution of the classification accuracy of different models simply in function of the batch size. We consider small batch sizes already as a factor of online class imbalance as not all classes can be present in the same batch.

We compare three methods: i) Tent without any class rebalancing method is used as baseline. ii) SAR [22] is not a class rebalancing method per se but the sample selection method introduced in this work is presented as a way to address the class imbalance problem by the authors. iii) DOT is an adaptation of the class-wise reweighting method proposed in [5] adapted to the context of test-time adaptation in [36]. The idea of DOT is to estimate the class frequencies in the test set by maintaining a momentum-based class-frequency vector $z \in \mathbb{R}^K$ where K is the total number of classes, based on the prediction of the model of each sample seen previously. At inference time, each new sample receives a weight in function of its pseudo label and the current z vector. A sample belonging to a rare class will receive a higher weight than a sample from a class seen more often. The DOT algorithm is detailed in the supplementary material.

Experimental results In Fig. 3, we can observe the following: i) On the ResNet50-BN architecture, the performance of all methods and for all batch sizes is dropping when the imbalance factor is increasing. Batch normalization does not seem to be a suitable normalization method when the test set is unbalanced ii) The performances of Tent and SAR are more stable when the imbalance factor varies on the ResNet50-GN architecture. On this architecture, DOT is the most performing method when the batch size is still high and the imbalance factor is still low. However, DOT performance is dropping drastically when the batch size becomes very small or the imbalance factor is very high. iii) Best performances are obtained by the VitBase-LN architecture. Performances are stable for all methods when the imbalance factor increases for a batch size of 16 or 8 but decrease when the imbalance factor increases for lower batch sizes. Our main takeaways from Fig. 3 are that group normalization and layer normalization are less sensitive than batch normalization to imbalance classes and that even if DOT and SAR are both performing better than Tent, the sample selection of SAR yields more stable performances in the case of small batch sizes and stronger class imbalance factor.

In Fig. 4, we observe that the performance of all methods on ResNet50-BN is dropping when the batch size decreases. On ResNet50-GN and VitBase-LN, the classification accuracy remains stable when the batch size decreases for all models, DOT yielding the best results except when the batch size is 1. This particular case is explained in the next paragraph. Our main takeaways from Fig. 4 are that architectures with group or layer normalization are more suitable to handle small batch sizes and that the class rebalancing method DOT is performing better than the sample selection method SAR for small batch sizes greater than 1.



(a) ResNet50-BN-16 (b) ResNet50-GN-16 (c) VitBase-LN-16

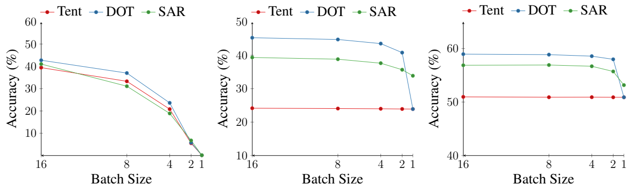
(d) ResNet50-BN-8 (e) ResNet50-GN-8 (f) VitBase-LN-8

(g) ResNet50-BN-4 (h) ResNet50-GN-4 (i) VitBase-LN-4

(j) ResNet50-BN-2 (k) ResNet50-GN-2 (l) VitBase-LN-2

(m) ResNet50-BN-1 (n) ResNet50-GN-1 (o) VitBase-LN-1

Figure 3. **Impact of Imbalance Factor, Architecture, and Batch Size on classification accuracy of different methods on ImageNet-C.** On ResNet50-BN, the performance of all models decreases when the imbalance factor increases. On ResNet50-GN, DOT, and SAR are more efficient than Tent, but SAR is more stable with very small batch sizes and stronger imbalance factors. On VitBase-LN, Tent performs lower than DOT and SAR with a batch size 4 and a moderate imbalance factor. However, DOT and SAR performance is dropping significantly for small batch sizes and strong imbalance factors. The number after the architecture in the legend is the batch size.



(a) ResNet50-BN (b) ResNet50-GN (c) VitBase-LN

Figure 4. **Impact of Architecture and Batch Size on the classification accuracy of different methods on ImageNet-C.** Batch-agnostic normalizations like group or layer normalization are more suitable to handle small batch sizes. Moreover, in this scenario, the class rebalancing method DOT is performing better than the sample selection method of SAR.

BatchSize=1	DOT	DOT+buff=2	DOT+buff=4	DOT+buff=8	DOT+buff=16
ResNet50-BN	0.14±0.00	**20.31**±0.02	20.31±0,02	20.31±0.02	20.31±0.02
ResNet50-GN	23.91±0,60	**38.94**±0.03	38.32±0,06	36.23±0.03	34.13±0.02
VitBase-LN	50.89±0.00	**54.15**±0.03	50.56±0,04	46.39±0.01	42.13±0.06

Table 1. **Impact of Additional Buffer on Tent performance on different architecture on ImageNet-C in the single point learning scenario.** An additional buffer of size 2 yields a significant performance improvement. Higher buffer sizes can lead to noisy sample weights and yield no additional improvement on ResNet50-BN or a performance decrease on ResNet50-GN and VitBase-LN.

Single point learning for DOT method In Fig. 4, we observe that in the specific case of batch size 1, the performance of DOT drops to the level of Tent. This is because in DOT, the weight of each sample in a batch is normalized by the sum of all weights of this batch. So, when the batch size is 1, the sum of the weights of the batch is equal to the weight of the single sample of the batch. Thus, the normalization of the weight of this single sample by the sum of all weights of the batch gives a weight of 1 and brings back to the same loss formulation as Tent. To address this issue, we propose to approximate the weight of a single sample in this particular case as if it was part of a bigger batch of size N. This approach does not require any additional processing time as we can still infer the class of an input test sample immediately and it is very cheap in terms of memory as we do not need to save any sample in a queue but just the weights of the N previous samples, which are only scalars. In Tab. 1, we analyze the impact of a buffer of different sizes on Tent performance on different architecture when the batch size is 1. We can see that an additional buffer of size 2 yields a significant performance improvement. Higher buffers yield no additional improvement on ResNet50-BN and a performance decrease on ResNet50-Gn and VitBase-LN. We assume that they lead to sample weights that are too noisy.

6. Sample selection

In the previous sections, we explored standard mechanisms to address covariate shift (through normalization) and label shift (through class rebalancing). In this section, we go one step further and explore mechanisms that cast TTA as a noisy learning problem. In particular, we explore the sample selection method first proposed in [21] and analyzed more thoroughly after in [22]. The main idea of this method is to select only reliable samples for the model adaptation.

Indeed, in [22], authors show that samples with high entropy are more likely to have a strong and noisy gradient potentially harmful to the model performance. Furthermore, low-entropy samples contribute more to the model adaptation than high-entropy ones. However, there is no easy way to directly filter out samples with a strong gradient from the optimization process. So, instead, an entropy-based filtering method was proposed. More precisely, a threshold entropy E_0 is defined as the maximum entropy $\log K$ multiplied by a factor F, which is a scalar with a value between 0 and 1, 1 meaning no selection at all. All samples with an entropy below this threshold $F \log K$ are kept whereas the others are discarded when computing the loss value to update the model. Formally, this filtering method can be expressed as a sample selection function S:

$$S(x) = \mathbb{I}_{\{E(x;\Theta)<E_0\}}(x) \tag{1}$$

where $\mathbb{I}_{\{.\}}(.)$ is an indicator function, $E(x;\Theta)$ is the entropy of sample x, and E_0 is a threshold predefined as:

$$E_0 = F \log K \tag{2}$$

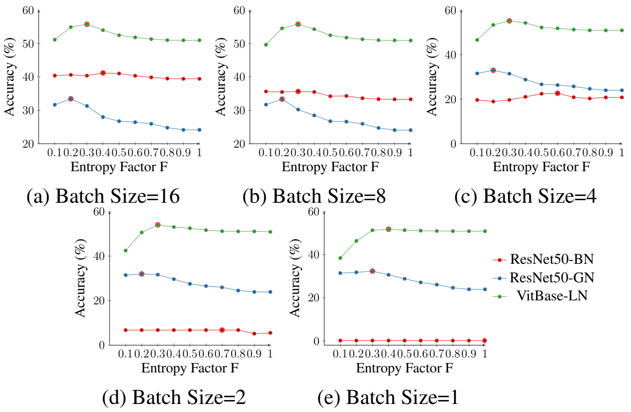
where K is the total number of classes in the dataset and F is a real number in $[0;1]$.

Experimental results In Fig. 5, we can see that fine-tuning the selection threshold via factor F can lead to a significant increase in the performances in all cases. We also observe that in the case of smaller batch sizes, the optimal value for F is smaller than the value of 0.5 recommended in [22] for a batch size of 64. Moreover, as mentioned in [22], another advantage of this method is that it requires less computational power to perform the adaptation as fewer samples are used in the optimization. *e.g.* for the Gaussian noise corruption, severity level 5, on ResNet50-GN and an entropy factor F of 0.4, the model forward passes 50K samples but keep less than 13K after selection for the backward pass, which is only 26% of the whole dataset.

7. Calibration

In this section, we investigate the problem of network calibration in the context of Fully TTA. The calibration of classification networks is a measure of the confidence of the predictions. It is of utmost importance in the context of Fully TTA as it impacts directly the predictions entropy. Temperature scaling is one technique introduced in [7] to improve the calibration of under- or overconfident neural networks by correcting the logits in the softmax function. Formally, it is expressed as:

$$softmax_{\tau}(z)_i = \frac{e^{z_i/\tau}}{\sum_{j=1}^{K} e^{z_j/\tau}} \tag{3}$$



(a) Batch Size=16 (b) Batch Size=8 (c) Batch Size=4

(d) Batch Size=2 (e) Batch Size=1

Figure 5. **Impact of Sample Selection and Architecture on classification accuracy of different methods on ImageNet-C**. The best results are circled in red. The optimal threshold varies in function of the architecture and the batch size and is lower for the smaller batch sizes than the values 0.5 or 0.4 for a batch size of 64 recommended in [21].

where τ is the temperature scaling factor, z is the logits vector of an input sample, i is a class index and K is the total number of classes. A τ value above 1 will lead to a higher entropy with a flattened distribution of the model predictions whereas a τ value smaller than 1 will lead to a low entropy with a more peaky predictions distribution. In the context of test-time adaptation, [6] shows that using temperature scaling improves the model accuracy after adaptation when using an entropy minimization-based method. [15] also shows that when meta-learning the optimal loss for test-time adaptation, the result is an entropy minimization loss with a temperature scaling factor. To determine the temperature scaling factor in our experiments, we follow [36] in the way to select hyperparameters using the 4 Imagenet-C validation corruptions. For each network architecture, we select the temperature scaling factor τ for each validation corruption using a grid search on values between 0.5 and 1.5 with a step of 0.1 and keep the average of the 4 values.

For the 3 network architectures considered, we obtain a temperature scaling factor of 1.2, which means that without correction, the models are too confident in their predictions.

Experimental results In Tab. 2, we observe that applying temperature scaling during adaptation leads to an increase in Tent performance on ResNet50-BN and VitBase-LN. On ResNet50-GN, the mean is slightly lower, but the standard deviation is significantly reduced, which means overall a better performance in terms of statistical significance. The performance increase is not very high when using temperature alone. However, we will see in Sec. 8 that it leads to higher performance when combined with other tricks.

	16	8	4	2	1
ResNet50-BN	39.43±0.13	33.30±0.04	20.81±0.08	5.53±0.01	0.14±0.00
ResNet50-BN+ temp	**39,45**±0,06	**33,86**±0,04	**20,84**±0,07	**6,11**±0,01	**0,15**±0,00
ResNet50-GN	**24,15**±0,55	**24,00**±0,54	**23,99**±0,56	**23,92**±0,57	**23,90**±0,58
ResNet50-GN+ temp	24,01±**0,17**	23,87±**0,17**	23,82±**0,15**	23,76±**0,19**	23,74±**0,19**
VitBase-LN	50,97±0,07	50,90±0,04	50,91±0,07	50,89±0,06	50,89±0.04
VitBase-LN + temp	**52,84**±0,27	**52,81**±0,26	**52,76**±0,26	**52,76**±0,20	**52,77**±0,22

Table 2. **Impact of Temperature on classification accuracy of Tent method performance on different architecture on ImageNet-C.** Using a temperature scaling factor increases the mean accuracy on ResNet50-BN and VitBase-LN. On ResNet50-GN, using temperature decreases slightly the mean classification accuracy but decreases also the standard deviation, which means that the model is better with respect to statistical significance.

8. Tricks combinations

In this section, we investigate the performance of Tent using different combinations of the tricks presented in the previous sections. For ResNet50-BN, we consider the usage of batch renormalization as an essential trick when dealing with very small batch sizes as presented in Sec. 4 and always integrate it in the different tricks combinations tested. In the ResNet50-BN section of Tab. 3, we report first the results already presented in Fig. 2 to see the performance improvement with batch renormalization. Then we consider all the possible combinations of 2 of the tricks presented and finally, we consider the combination of all the tricks. For ResNet50-GN and VitBase-LN, we also present results considering all the possible combinations of 2 of the tricks presented previously and then combining all the tricks.

Experimental results In Tab. 3, we observe that when using a ResNet50-BN network, the best pair of tricks is the class rebalancing method DOT combined with the entropy-based sample selection. The best results overall are obtained when using this pair with a temperature scaling factor, in other words when using all tricks together. In this case, compared to Tent, we obtain an average improvement of +17.08% accuracy over all batch sizes. In the case of a ResNet50-GN architecture, the best pair of tricks is class rebalancing combined with the temperature scaling factor. Surprisingly, combining temperature scaling with sample selection is performing better than vanilla Tent but much lower than other pairs of tricks. We assume that as the temperature scaling is changing the entropy of the test samples, a finer tuning of the sample selection margin should be done to ensure that samples useful for the model adaptation are not discarded. The best performances are obtained using all tricks. In this case, we obtain an average improvement of +19.92% accuracy over all batch sizes compared to Tent. When considering the VitBase-LN architecture, we can see that the two pairs of tricks class rebalancing and temperature and class rebalancing and sample selection are close over all the batch sizes and yield the best results of the pairs of tricks. The overall best results are obtained when combining all tricks. Doing this leads to an average improvement compared to Tent of +7.66% over all batch sizes. Our main takeaway for this series of experiments is that the best results are obtained when combining all tricks (class rebalancing, sample selection, and temperature scaling), and this for the 3 architectures and the different batch sizes considered. Among the different architectures, VitBase-LN has the best classification accuracy when combining all the tricks and on all the batch sizes tested.

	Tent +				Batch Size				
	BR	CR	SS	T	16	8	4	2	1
ResNet50-BN					39.40±0.13	33.30±0.04	20.81±0.08	5.53±0.01	0.14±0.00
	✓				43.26±0.01	41.39±0.06	37.72±0.05	30.84±0.04	20.25±0.01
	✓	✓		✓	45.89±0.06	43.70±0.05	39.17±0.05	31.44±0.04	**20.31**±0.02
	✓		✓	✓	45.17±0.26	43.03±0.11	39.02±0.07	31.60±0.05	20.26±0.01
	✓	✓	✓		46.57±0.07	44.46±0.01	39.95±0.01	31.65±0.01	20.30±0.02
	✓	✓	✓	✓	**46.90**±0.12	**44.90**±0.09	**40.42**±0.14	**32.03**±0.05	**20.31**±0.02
ResNet50-GN					24.15±0.55	24.06±0.54	23.99±0.57	23.92±0.57	23.90±0.58
		✓		✓	46.35±0.07	45.89±0.09	44.77±0.01	42.07±0.03	39.31±0.64
			✓	✓	26.85±0.17	27.34±0.55	29.03±0.59	30.19±0.20	27.20±0.48
		✓	✓		45.78±0.09	45.31±0.11	44.21±0.01	41.33±0.01	38.94±0.03
		✓	✓	✓	**46.50**±0.05	**46.07**±0.08	**45.02**±0.01	**42.32**±0.01	**39.70**±0.04
VitBase-LN					50.97±0.07	50.90±0.04	50.91±0.07	50.89±0.06	50.89±0.04
		✓		✓	59.26±0.03	59.20±0.02	58.97±0.04	58.52±0.05	54.68±0.03
			✓	✓	57.59±0.44	58.11±0.14	57.88±0.09	57.02±0.10	55.10±0.07
		✓	✓		59.31±0.06	59.22±0.04	58.96±0.00	57.51±0.78	54.15±0.03
		✓	✓	✓	**59.80**±0.07	**59.77**±0.04	**59.59**±0.03	**59.04**±0.06	**55.15**±0.03

BR=BatchRenorm, T=Temperature, CR=Class Rebalancing, SS=Sample Selection

Table 3. **Effect of Tricks Combination on model performance.** Best results are obtained when combining all tricks and this for the 3 architectures and the different batch sizes considered. Among the different architectures, VitBase-LN has the best classification accuracy in all the different setups.

9. Comparison to other methods and on other datasets

In this final experimental section, we compare the performance of BoT (i.e. Tent with all the tricks presented in this article) to a vanilla Tent and 2 state-of-the-art methods, SAR [22] and Delta [36]. This comparison is performed on different network architectures and different datasets: ResNet50-BN, ResNet50-GN, VitBase-LN for ImageNet-C, ImageNet-Rendition and ImageNet-Sketch, and ResNet101 for VisDA2017.

Experimental results In Tab. 4, we can see that on the ImageNet-C dataset, BoT obtains better results than a vanilla Tent, and the two state-of-the-art methods for all the batch sizes considered. Interesting to see is the collapse of SAR performance for very small batch sizes (2 and 1) on ResNet50-BN that we do not observe with Delta due to the usage of batch renormalization. If the performance increase by using all the tricks is not significant on ResNet50-BN (+0.78% accuracy on average versus Delta), it is much more noticeable on ResNet50-GN (+4.31% ac-

	Method	Batch Size				
		16	8	4	2	1
ResNet50-BN	Tent	39.43±0.13	33.30±0.04	20.81±0.08	5.53±0.01	0.14±0.00
	SAR	41.02±0.29	31.10±0.08	18.90±0.04	6.78±0.00	0.14±0.00
	Delta	46.33±0.78	43.67±0.05	39.16±0.04	31.26±0.05	20.25±0.01
	BoT	**46.90**±0.1	**44.90**±0.09	**40.42**±0.14	**32.03**±0.05	**20.31**±0.02
ResNet50-GN	Tent	24.15±0.55	24.05±0.54	23.99±0.57	23.92±0.57	23.90±0.58
	SAR	39.32±0.17	38.80±0.14	37.61±0.39	35.66±0.28	33.86±0.06
	Delta	45.22±0.06	44.70±0.09	43.47±0.02	40.77±0.01	23.91±0.60
	BoT	**46.50**±0.05	**46.07**±0.08	**45.02**±0.01	**42.32**±0.01	**39.70**±0.04
VitBase-LN	Tent	50.97±0.07	50.90±0.04	50.91±0.07	50.90±0.06	50.89±0.04
	SAR	56.87±0.15	56.92±0.10	56.69±0.13	55.71±0.16	53.16±0.16
	Delta	58.95±0.05	58.86±0.04	58.57±0.03	57.98±0.04	50.89±0.04
	BoT	**59.80**±0.07	**59.77**±0.04	**59.59**±0.03	**59.04**±0.06	**54.68**±0.03

Table 4. **Results on ImageNet-C.** BoT obtains better results than Tent and the 2 state-of-the-art methods in all cases. If the performance increase of BoT is not significant on ResNet50-BN (+0.78% accuracy in average versus Delta), it is much more noticeable on ResNet50-GN (+4.31% accuracy in average versus Delta) and VitBase-LN (+1.53.31% accuracy in average versus Delta).

	Method	Batch Size				
		16	8	4	2	1
ResNet50-BN	Tent	40.80±0.11	37.75±0.12	29.70±0.21	14.24±0.05	0.56±0.00
	SAR	42.11±0.10	38.95±0.21	30.07±0.05	16.13±0.12	0.57±0.00
	Delta	43.11±0.15	41.80±0.23	39.64±0.16	35.17±0.06	26.75±0.01
	BoT	**44.68**±0.24	**43.12**±0.11	**40.61**±0.22	**35.55**±0.04	**26.75**±0.00
ResNet50-GN	Tent	39.35±0.16	39.29±0.18	39.28±0.19	39.27±0.18	39.26±0.18
	SAR	42.94±0.108	42.75±0.05	42.28±0.09	41.75±0.06	41.84±0.05
	Delta	43.10±0.05	43.11±0.05	42.74±0.12	41.89±0.10	42.18±0.03
	BoT	**44.21**±0.06	**44.18**±0.10	**43.84**±0.20	**42.96**±0.16	**42.49**±0.08
VitBase-LN	Tent	43.28±1.04	42.81±1.04	42.48±0.87	42.28±1.05	42.49±1.32
	SAR	52.72±0.19	52.59±0.25	52.20±0.16	50.92±0.11	49.95±0.18
	Delta	53.32±0.23	53.31±0.28	53.03±0.24	52.25±0.34	49.76±0.20
	BoT	**54.63**±0.18	**57.74**±0.19	**54.62**±0.25	**53.86**±0.28	**51.91**±0.15

Table 5. **Results on ImageNet-Rendition.** The performance increase of BoT compared to Delta is similar on ResNet50-BN and ResNet50-GN (respectively +0.85% and +0.87% accuracy) but reaches +1.23% accuracy on VitBase-LN.

curacy on average versus Delta) and VitBase-LN (+1.53% accuracy in average versus Delta). In Tab. 5, we also observe that BoT performs the best in all cases. Interesting to note is that results are more stable over the different batch sizes with ResNet50-GN compared to ResNet50-BN, which is in line with observations from previous experiments. Delta performs better than SAR but worse than BoT. The performance increase of BoT compared to Delta is similar on ResNet50-BN and ResNet50-GN (respectively +0.85% and +0.87% accuracy) but reaches +1.23% accuracy on VitBase-LN. In Tab. 6, we make the same observations on ImageNet-Sketch as on the other ImageNet variants. ResNet50-BN performance drops when the batch size becomes small. In all cases, Delta performs better than SAR but not as good as BoT. BoT performs best in all cases. The performance increase of BoT versus Delta is +0.72% accuracy on ResNet50-BN, +1.32% accuracy on ResNet50-GN, and +1.03% accuracy on VitBase-LN. In Tab. 7, we observe that also for the VisDA2017 dataset, results are in line with previous experiments. Delta performs better than Tent and SAR but not as well as BoT. The performance improvement of BoT versus Delta is +0.36% accuracy on ResNet101.

	Method	Batch Size				
		16	8	4	2	1
ResNet50-BN	Tent	27.82±0.30	22.47±0.40	10.71±0.42	2.94±0.08	0.13±0.00
	SAR	31.05±0.29	26.73±0.20	16.80±0.07	6.72±0.05	0.13±0.00
	Delta	31.92±0.11	30.36±0.16	27,32±0.16	22.56±0.16	**15.58**±0.04
	BoT	**33.24**±0.13	**31.50**±0.21	**28.16**±0.12	**22.86**±0.16	**15.58**±0.04
ResNet50-GN	Tent	23.04±0.40	22.95±0.38	22.93±0.38	22.92±0.38	22.92±0.35
	SAR	32.11±0.50	32.26±0.07	31.89±0.16	31.16±0.20	31.64±0.25
	Delta	34.50±0.20	34.26±0.09	33.57±0.18	31.56±0.08	30.93±0.07
	BoT	**35.77**±0.03	**35.49**±0.19	**34.91**±0.15	**33.19**±0.10	**32.07**±0.09
VitBase-LN	Tent	5.83±0.32	5.69±0.43	5.59±0.44	5.38±0.28	5.51±0.49
	SAR	25.40±0.65	25.88±0.64	27.87±0.08	32.89±0.57	30.68±0.99
	Delta	38.67±0.08	38.50±0.08	38.18±0.11	37.18±0.14	33.90±0.08
	BoT	**39.69**±0.06	**39.68**±0.06	**39.50**±0.09	**38.64**±0.03	**34.09**±0.10

Table 6. **Results on ImageNet-Sketch.** BoT performs best in all case. The performance increase of BoT versus Delta is +0.72% accuracy on ResNet50-BN, +1.32% accuracy on ResNet50-GN and +1.03% accuracy on VitBase-LN.

	Method	Batch Size				
		16	8	4	2	1
ResNet101	Tent	65.30±0.08	64.65±0.18	63.47±0.12	58.89±0.33	49.10±0.04
	SAR	63.08±0.03	57.47±0.05	46.20±0.09	24.81±0.16	18.63±0.01
	Delta	73.20±0.08	71.52±0.12	68.16±0.11	61.41±0.20	49.08±0.03
	BoT	**73.54**±0.09	**71.70**±0.07	**68.17**±0.19	**61.49**±0.10	**50.28**±0.09

Table 7. **Results on VisDA2017.** Delta performs better than Tent and SAR but not as good as BoT. The performance improvement of BoT versus Delta is +0.36% accuracy and +4.75% versus Tent on ResNet101.

10. Conclusion

In this work, we addressed the Fully Test-Time Adaptation problem when dealing with small batch sizes by analyzing the following tricks and methods: i) Usage of Batch renormalization or batch-agnostic normalization ii) Class re-balancing iii) Entropy-based sample selection iv) Temperature scaling. Our experimental results show that if those tricks used alone already yield an improved classification accuracy, using them in pairs is even better, and the best results are obtained by combining them all. By doing that, we significantly improve the current state-of-the-art across 4 different image datasets in terms of prediction performances. Furthermore, the selected tricks bring additional benefits concerning the computational load: i) Using group normalization instead of batch normalization in ResNet50 yields more stable results for the same number of total parameters ii) using the entropy-based sample selection improves the adapted model performance by using fewer samples. We hope that this study will be useful for the community and that the presented tricks and techniques will be integrated into future baselines and benchmarks.

11. Acknowledgment

This research was supported by the National Science and Engineering Research Council of Canada (NSERC), via its Discovery Grant program, and enabled in part by support provided by Calcul Québec and the Digital Research Alliance of Canada .

References

[1] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv: 1607.06450*, 2016. 3

[2] Alexander Bartler, Andre Bühler, Felix Wiewel, Mario Döbler, and Bin Yang. Mt3: Meta test-time training for self-supervised test-time adaption. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022. 1

[3] Malik Boudiaf, Tom Denton, Bart van Merriënboer, Vincent Dumoulin, and Eleni Triantafillou. In search for a generalizable method for source free domain adaptation. *International Conference on Machine Learning (ICML)*, 2023. 2, 3

[4] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[5] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. Class-balanced loss based on effective number of samples. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[6] Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and Zico Kolter. Test-time adaptation via conjugate pseudo-labels. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 6

[7] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017. 6

[8] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3

[9] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *International Conference on Computer Vision (ICCV)*, 2021. 1, 3

[10] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[11] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 3

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 3

[13] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[14] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021. 1

[15] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *International Conference on Machine Learning (ICML) Workshop on challenges in representation learning*, 2013. 2, 6

[16] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[17] Jian Liang, D. Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning (ICML)*, 2020. 2

[18] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. Ttn: A domain-shift aware batch normalization in test-time adaptation. In *International Conference on Learning Representations (ICLR)*, 2023. 1, 2, 3

[19] M. Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: Dynamic unsupervised domain adaptation by normalization. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[20] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *International Conference on Machine Learning (ICML) Uncertainty and Robustness in Deep Learning Workshop*, 2020. 2

[21] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shi Dong Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning (ICML)*, 2022. 2, 5, 6

[22] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations (ICLR)*, 2023. 1, 2, 3, 4, 5, 6, 7

[23] David Osowiechi, Gustavo Adolfo Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ismail Ben Ayed, and Christian Desrosiers. Tttflow: Unsupervised test-time training with normalizing flow. *Winter Conference on Applications of Computer Vision (WACV)*, 2023. 2

[24] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv: 1710.06924*, 2017. 3

[25] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. Dataset shift in machine learning. Yale University Press in association with the Museum of London, 2009. 1

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Chal-

lenge. *International Journal of Computer Vision (IJCV)*, 2015. 3

[27] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[28] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020. 1, 2

[29] Yushun Tang, Ce Zhang, Heng Xu, Shuoshuo Chen, Jie Cheng, Luziwei Leng, Qinghai Guo, and Zhihai He. Neuro-modulated hebbian learning for fully test-time adaptation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[30] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 2, 3

[31] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3

[32] Shuo Wang, Leandro L. Minku, and Xin Yao. Dealing with multiple classes in online class imbalance learning. In *International Joint Conference on Artificial Intelligence*, 2016. 1, 4

[33] Yuxin Wu and Kaiming He. Group normalization. *International Journal of Computer Vision*, 2018. 3

[34] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3

[35] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2, 3

[36] Bowen Zhao, Chen Chen, and Shutao Xia. Delta: degradation-free fully test-time adaptation. In *International Conference on Learning Representations (ICLR)*, 2023. 1, 2, 3, 4, 6, 7