Exploring viability of Test-Time Training: Application to 3D segmentation in Multiple Sclerosis

Benoît Gérin*[1] Maxime Zanella*[1,2] Maxence Wynen[1,3]
Saïd Mahmoudi[2] Benoît Macq[1] Christophe De Vleeschouwer[1]
[1]ICTEAM, UCLouvain, Belgium [2]ILIA, UMons, Belgium [3]NIL, UCLouvain, Belgium

***Abstract*—Test-Time Training (TTT) is an unsupervised domain adaptation technique employing a self-supervised task performed by an attached branch model. While justification of key design choices are often neglected in the literature, we explore the viability of TTT in a real-case scenario and conduct an extensive evaluation of key hyperparameters in TTT, including the choice and the placement of the auxiliary task, the type of normalization layer, and the model parameters to adapt. We carry out this study in Multiple Sclerosis (MS) diagnosis relying on focal lesions visible in conventional MRI. Manual lesion segmentation based on automated methods face significant challenges in clinical integration due to MRI domain shifts, i.e. discrepancies between training and deployment data, notably due to different acquisition settings. We apply TTT to each patient individually, offering an effective strategy to mitigate domain shift without the need of additional annotated data or data from other patients. We ground our experiments on real-world distribution shifts using three distinct MS datasets. Finally, we propose general guidelines to apply TTT in practice. Code available at github.com/Gerin-Benoit/ttt-for-multi-sclerosis.**

***Index Terms*—test-time training, unsupervised domain adaptation, multiple sclerosis, lesion segmentation, 3D MRI**

I. Introduction

Recently introduced, **Test-Time Training (TTT)** is a subfield of domain adaptation, which is a promising research direction that studies solutions to adapt a model trained on a source (i.e., in-distribution) dataset in order to improve its performance on a target (i.e., out-of-distribution) dataset [1]. TTT addresses this specific question by using an auxiliary task in a self-supervised manner [2] to gain information on the target distribution. TTT has proven to bring significant improvement on numerous tasks without expensive labeling procedure requirement [2]–[6]. While simple in essence, the practical implementation of TTT is hindered by the difficult task of hyperparameters selection, especially when no validation set is available.

These domain adaption challenges arise particularly in **Multiple Sclerosis (MS)** lesion segmentation [7]–[12]. MS is a common neurodegenerative disease [13] characterized by focal lesions in the central nervous system, mainly in the white matter, grey matter and spinal cord [14]. Detected through magnetic resonance imaging (MRI), these lesions play a crucial role in the current criteria for MS diagnosis [14] and prognosis [15]. Historically, expert neuroradiologists have been entrusted with the complex task of MS lesion segmentation, an approach that is not only labor-intensive but also prone to errors [16]. While numerous automated methods have emerged to assist experts in this task [17], [18], their use in everyday clinical practice remains limited [19]. One major obstacle is domain shifts — discrepancies between training and deployment data — that often negatively affect model performance [20]. In the context of MS lesion segmentation, these shifts primarily arise due to variations in data acquisition parameters [9], [11], [12], [21]. Still, TTT in MS lesion segmentation remains unexplored despite the potential observed in other application fields. We aim to fill this gap and investigate the viability of TTT in MS lesion segmentation using Fluid Attenuated Inversion Recovery (FLAIR) images. We conduct this study on a per-patient basis to align with the clinical

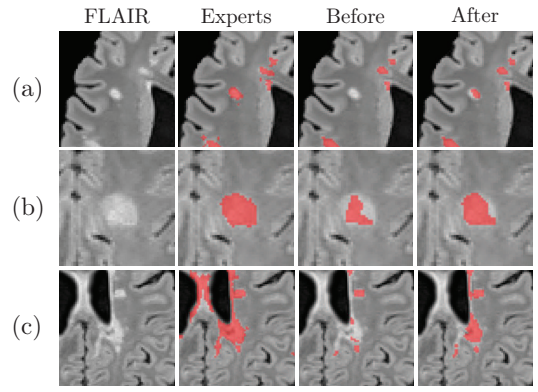


Fig. 1: Comparative magnified views of FLAIR images and segmentation masks. From left to right: original FLAIR, segmentation by expert consensus, model output before test-time training (TTT), and after TTT. All segmentations are highlighted in red. We can observe that TTT helps (a) finding missed lesions, (b) enhancing the contour of lesions or (c) improving the coverage of complex regions.

* Equal contribution and corresponding authors: {benoit.gerin, maxime.zanella}@uclouvain.be

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00110

practice of individualized patient care. Some benefits of such methods are depicted on Figure 1 where results before and after using our solution are compared.

The contributions of this paper can be summarized as follows: **(i)** the application of the TTT paradigm to real-world domain shifts occurring in medical imaging, **(ii)** a comparative study of several auxiliary tasks within the TTT framework for MS lesion segmentation, **(iii)** a comprehensive examination of TTT hyperparameters, which are often either arbitrarily chosen or lack robust empirical grounding, and **(iv)** guidelines to apply TTT in MS lesion segmentation.

II. Related Work

Unsupervised domain adaptation aims to enhance model performance when faced with domain shifts. Conventional approaches often focus on aligning feature distributions using discrepancy loss mechanisms [22] or through adversarial training techniques [23]. A more recent work proposes to freeze the classifier head while encouraging the model to produce source-like, or more confident, predictions [24].

Test-time adaptation offers a practical setting where the source data is not accessible during testing. A strong baseline is to rely solely on the adaptation of the batch normalization layer statistics without additional training [25]. Another common method introduces an additional entropy minimization term, forcing the model towards more confident predictions [26]. This approach can be extended to segmentation, for example, by incorporating shape descriptors to guide the adaptation process [27].

Test-time training incorporates an auxiliary task, such as rotation prediction, to guide model adaptation [2]. Subsequent work gives insights on the importance of aligning the auxiliary task with the primary one [3]. Various auxiliary tasks have been proposed, including input reconstruction with a masked autoencoder [4], feature distribution realignment via normalizing flows [5], and meta-learning combined with a contrastive loss [6].

TTT in 3D segmentation is an emerging area of study. It has been used to improve knowledge transfer from synthetic CT images through reconstruction-based tasks [28]. Additionally, it can facilitate adaptation from different scanners and protocols by leveraging transferable spatial relations inherent to the human body structure [29], or by artificially degrading segmentation masks during training to produce more plausible predictions at inference [30].

While previous works in the literature propose highly tuned methods for specific cases, there is no clear study on how to chose hyperparameters to deploy TTT in practice: the branch position can be located at the beginning or in the middle of the model [2]–[6], [28]–[30], the parameters to adapt are either the affine parameters of the normalization layer or all the parameters [2], [3], [5], [26]–[29], and the number of iterations before stopping the adaptation is not precisely discussed [4], [5], [28]. In our work, we carefully address these flaws by making an extensive study on the key design choices of TTT and highlighting the performance gain brought by TTT on a real domain shift scenario.

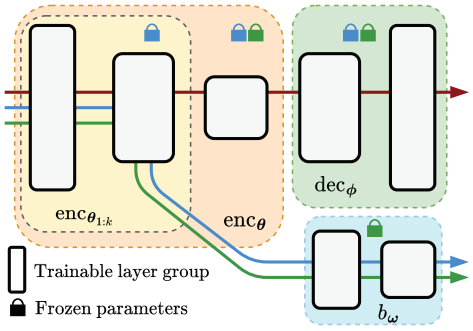


Fig. 2: Training and adaptation pipeline. Arrows indicate forward flow for core model training (red), branch training (blue) and test-time training (green). For each case, locks indicate which layers are not modified. Best view in color.

III. Method

This section formally describes our test-time training procedure. A core model is trained on a prediction task (e.g., semantic segmentation) and a second model — the branch — learns to solve an auxiliary task. During TTT, the branch model is used to adapt the core model.

Core model training (Source). Let the core model be the concatenation of a feature encoder $\text{enc}_{\boldsymbol{\theta}}$ and a decoder $\text{dec}_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ respectively. The encoder parameters are the union of K successive groups of layer parameters $\{\theta_1, \theta_2, \ldots, \theta_K\}$. The k^{th} latent representation $\mathbf{z}_k = \text{enc}_{\boldsymbol{\theta}_{1:k}}(\mathbf{x})$ of the input volume batch $\mathbf{x}$ is extracted with the concatenation of the k-first layer groups of $\text{enc}_{\boldsymbol{\theta}}$. The core model prediction is denoted $\hat{\mathbf{y}} = \text{dec}_{\boldsymbol{\phi}} \circ \text{enc}_{\boldsymbol{\theta}}(\mathbf{x})$.
$\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are trained by minimizing a supervision error function l for n volume batches $\{\mathbf{x}_i\}_{i=1}^n$ and labels $\{\mathbf{y}_i\}_{i=1}^n$:

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \frac{1}{n} \sum_{i=1}^{n} l(\mathbf{x}_i, \mathbf{y}_i; \boldsymbol{\theta}, \boldsymbol{\phi}) \tag{1}$$

Branch training (Source). We plug in an auxiliary branch $b_{\boldsymbol{\omega}}$ with parameters $\boldsymbol{\omega}$, which is designed to solve an auxiliary task. This so-called branch model is connected to the k^{th} encoder layer and outputs auxiliary prediction $\hat{\mathbf{y}}^{aux} = b_{\boldsymbol{\omega}} \circ \text{enc}_{\boldsymbol{\theta}_{1:k}}(\mathbf{x})$. The branch parameters are trained by minimizing the auxiliary task error l^{aux} while keeping the encoder parameters frozen:

$$\min_{\boldsymbol{\omega}} \frac{1}{n} \sum_{i=1}^{n} l^{aux}(\mathbf{x}_i, \mathbf{y}_i^{aux}; \boldsymbol{\theta}_{1:k}, \boldsymbol{\omega}) \tag{2}$$

Test-time training (Target). For each patient, the branch is used to retrain some parameters to produce a new latent representation $\mathbf{z}_k$ by solving the auxiliary task. Following previous works [5], [24], [28], [29], only the k-first group

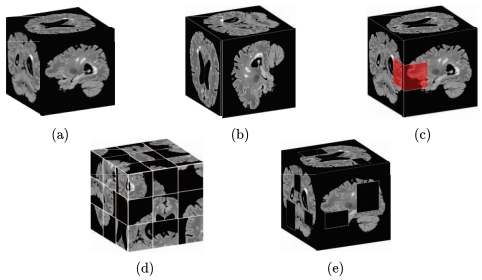


Fig. 3: Illustration of the auxiliary tasks: (a) original patch; (b) rotation prediction; (c) relative patch location; (d) jigsaw; (e) inpainting. One of these auxiliary tasks is performed by the branch.

of layers are adapted by minimizing the auxiliary task loss l^{aux}:

$$\min_{\boldsymbol{\theta}_{1:k}} \frac{1}{n} \sum_{i=1}^{n} l^{aux}(\mathbf{x}_i, \mathbf{y}_i^{aux}; \boldsymbol{\theta}, \boldsymbol{\omega}) \tag{3}$$

where the parameters $\boldsymbol{\theta}_{k+1:K}$, $\boldsymbol{\phi}$ and $\boldsymbol{\omega}$ are kept frozen during the adaptation. For an overview of our pipeline, we refer the reader to Figure 2.

IV. Experimental setting

Datasets. Based on the Shifts 2.0 challenge [21], we use three datasets (namely `best`, `msseg` and `ljubljana`) originating from five acquisition centers across six MRI scanners with varying spatial resolutions. We alternatively regard one dataset as source for core model and branch training, while the others are considered as targets for TTT evaluation.
In our experiments, a *partition* is defined as a source dataset and two target (i.e., out-of-distribution) datasets. Each target dataset is further divided into a validation set of 10 patients and a test set including the remaining patients. This procedure leads to three distinct *partitions*, one per source dataset.

Auxiliary tasks. We study four different self-supervised tasks, summarized on Figure 3.

- **Rotation prediction** (Fig. 3b) consists in predicting the angle of a randomly rotated input patch. We choose the hyperparameters of this task following [31].
- **Relative patch location** (Fig. 3c) divides a patch into a grid, wherein a sub-volume is randomly selected. This task consists of predicting the location of the selected sub-volume with respect to the center sub-volume. We choose the hyperparameters of this task following [31].
- **Jigsaw** (Fig. 3d) divides a patch into a grid and shuffles the created sub-volumes according to a randomly selected permutation from a predefined set. The objective of the task is to determine which of these permutations was applied. We choose the hyperparameters of this task following [31].
- **Inpainting** (Fig. 3e) consists in reconstructing a randomly masked input patch. We choose the hyperparameters of this task following [32].

Evaluation protocol. We assess each combination of auxiliary task and set of hyperparameters on each *partition* with the three following main steps:

1) We first train a core model, then a branch model on the source dataset.
2) For a given source, the key hyperparameters — namely, learning rate, number of iterations, parameters to adapt — are determined with the target validation sets. This approach is more clinically realistic: individually tuning TTT hyperparameters for each patient would be impractical in the absence of ground truth data.
3) We evaluate on the corresponding test sets of the current *partition*. Average Dice Similarity Coefficient (DSC) across test sets is reported accordingly.

Implementation details. Inspired by the work of [33], we opt for a 3D UNet as our core model architecture, which is widely employed in the literature [18]. Its encoder consists of three successive trainable layer groups corresponding to $k = 1, 2$ and 3 in Equation 3. We also compare two normalization layer types for the core model: Batch Normalization (BN) which is often considered in domain adaptation settings [25], and Instance Normalization (IN) [34]. The branch is composed of a series of upsampling convolutional layers for inpainting, or a convolutional projector to reduce feature dimensions followed by three fully connected layers for the other tasks.
The training of each model is performed for a maximum of 300 epochs with a batch size of 4. Patches of size 96^3 are cropped in FLAIR images in the same fashion as [21]. The core model and the branch are trained separately using Adam optimizer with a cosine scheduler. The learning rate is decreased from 5e-4 to 1e-5, or 1e-4 to 1e-5 depending on the best performing model or branch. Test-time training is performed for 150 epochs with Adam optimizer at constant learning rate among 7 possible values ranging from 1e-5 to 1e-2 equally spaced in log-scale depending on the best validation DSC.

Study scope. We investigate several aspects of the test-time training framework. Specifically, our study focuses on:

- The type of **normalization layer** to use in the core model (BN or IN).
- The **location of the branch** model (at level $k = 1, 2$ or 3).
- The **subset of parameters** in the core model to adapt (either all preceding parameters leading up to the branch or solely the preceding affine parameters of the normalization layers).
- The type of **auxiliary task** to solve (rotation prediction, relative patch location, jigsaw, or inpainting).
- The number of adaptation **iterations** to perform (from 0 to 150).

		Source			
		Self	best	msseg	ljub.
Target	best	0.722	0.0%	-18.2%	-19.9%
	msseg	0.726	-26.1%	0.0%	-9.0%
	ljub.	0.616	-25.4%	-3.7%	0.0%

(a) Batch Normalization.

		Source			
		Self	best	msseg	ljub.
Target	best	0.736	0.0%	-18.8%	-21.5%
	msseg	0.763	-51.6%	0.0%	-13.9%
	ljub.	0.659	-43.8%	-4.8%	0.0%

(b) Instance Normalization.

TABLE I: Cross-dataset relative degradation of DSC performance. The **Self** column reports DSC value of the core models on their own source test set. For each cell (i, j), relative degradation of core model trained on source dataset j is compared to performance of core model trained on dataset i. For example, core model with BN trained on best suffers a decrease to $0.616 * (100 - 25.4\%) = 0.460$ DSC when evaluated on ljubljana with respect to the performance of the core model trained on ljubljana.

Partition	best → targets						msseg → targets						ljubljana → targets					
Train. parameters	Affine			All			Affine			All			Affine			All		
k (branch level)	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
No adapt. (DSC)	0.498						0.592						0.619					
BNA [25]	+1.0%	+1.8%	+2.0%	+1.0%	+1.8%	+2.0%	+0.7%	+0.7%	+0.7%	+0.7%	+0.7%	+0.7%	-0.2%	-0.2%	-0.3%	-0.2%	-0.2%	-0.3%
TENT [26]	+4.8%	+8.2%	+8.8%				+1.0%	+2.5%	+2.5%				-1.9%	-0.2%	-0.2%			
TTT w/ Rotation	+3.4%	+10.0%	+8.6%	-22.7%	-99.4%	+9.2%	+2.2%	+2.5%	-	+0.7%	-5.9%	-	-1.9%	-1.5%	0.0%	-1.6%	-11.0%	0.0%
TTT w/ RPL	+9.4%	+11.4%	+10.8%	+13.5%	+11.8%	+9.8%	-0.2%	+0.7%	-	-96.1%	-83.4%	-	-3.9%	-0.3%	0.0%	-98.7%	-2.6%	0.0%
TTT w/ Jigsaw	+6.2%	+6.8%	+7.4%	+12.9%	+5.4%	+6.8%	+1.4%	+2.9%	-	+1.7%	+2.0%	-	-2.3%	0.0%	0.0%	-0.6%	0.0%	0.0%
TTT w/ Inpainting	-1.2%	+5.0%	+13.1%	-4.6%	+11.8%	+6.6%	+1.4%	+4.7%	+2.7%	+1.7%	+5.4%	+4.1%	+1.1%	0.0%	0.0%	-2.4%	0.0%	0.0%

(a) Batch Normalization.

Partition	best → targets						msseg → targets						ljubljana → targets					
Train. parameters	Affine			All			Affine			All			Affine			All		
k	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
No adapt. (DSC)	0.370						0.613						0.618					
TENT [26]	+4.1%	+4.1%	+4.1%				0.0%	0.0%	-0.2%				0.0%	0.0%	0.0%			
TTT w/ Rotation	+24.9%	+16.8%	+19.7%	-97.8%	-80.3%	-98.4%	-3.3%	0.0%	+1.1%	+1.0%	-32.3%	-90.2%	0.0%	-0.2%	-	+0.2%	-4.4%	-
TTT w/ RPL	+50.3%	+41.4%	+31.9%	-96.5%	+27.0%	+50.0%	+0.8%	+2.0%	-0.1%	-88.7%	-90.9%	-90.7%	-0.8%	0.0%	-	-2.6%	+0.5%	-
TTT w/ Jigsaw	+51.1%	+51.9%	-	+34.3%	+24.9%	-	0.0%	0.0%	-	+0.8%	0.0%	-	-2.4%	-2.8%	-	0.0%	0.0%	-
TTT w/ Inpainting	+52.2%	+30.5%	+47.3%	+20.5%	+25.4%	+32.4%	0.0%	+0.2%	+0.5%	0.0%	-0.7%	+0.35%	+0.5%	+0.6%	+2.6%	+0.6%	-0.2%	+0.2%

(b) Instance Normalization.

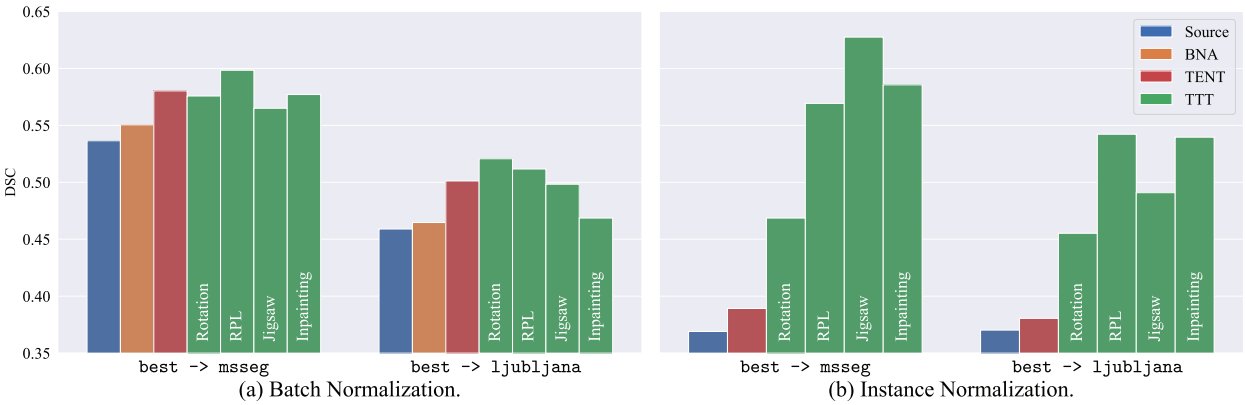
TABLE II: Summary of results for each method with best hyperparameters on validation set. Relative improvement in comparison to baseline (in gray) is reported in percentage (%). Green indicates improvement and red degradation, with intensity scale proportional to significance. ”-” corresponds to inability to train the branch in the hyperparameters range.

V. Results and Discussion

This section aims at presenting the results across the domain shifts induced by the three *partitions*. We follow the Evaluation protocol and report the DSC for the best-performing hyperparameters on the validation set. We compare with the adaptation of BN statistics only (BNA) [25] and entropy minimization (TENT) [26] methods. Qualitative results are shown on Figure 1.

Domain shift. As shown in Table I, baseline models suffer DSC drops when tested on non-source datasets, indicating domain shifts between source and target distributions. This is the case for every *partition* and especially for best→ targets. Moreover, models trained on ljubljana or msseg are more robust when evaluated on msseg or ljubljana, respectively, suggesting that the knowledge transfer between these two datasets is less impacted by the domain shift. Finally, Table II shows that TTT applied on a per-patient basis effectively improves model performance when the domain shift is significant.

Hyperparameters. Table I indicates that core models with BN layers perform more robustly without adaptation compared to their IN counterparts, which was first proposed to preserve instance-specific mean and covariate shift [34]. However, after adaptation, both types of normalization layer achieve similar DSC performance on the target datasets. In Table II, the best-performing configuration for *partition* 1 shows a relative improvement in DSC of +13.1% for BN and +52.2% for IN. For *partition* 2, the improvement is +5.4% for BN and +2% for IN. No significant improvement is observed for *partition* 3. This suggests that the prevailing consensus in the literature,



(a) Batch Normalization. (b) Instance Normalization.

Fig. 4: Performance of application of TTT on core model trained on `best` with the different auxiliary tasks compared to model without adaptation (Source), BNA [25] and TENT [26] for (a) BN core model and (b) IN core model. Hyperparameters choice follows the proposed guidelines (1, 3, 4).

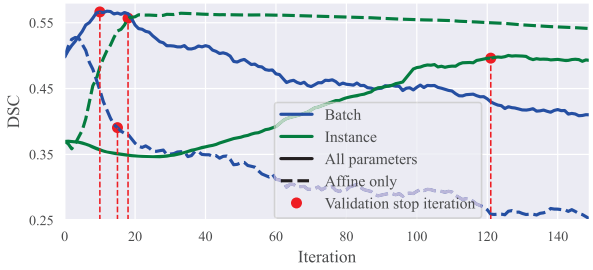


Fig. 5: `best` $\rightarrow$ targets *partition* performances for jigsaw auxiliary task and $k = 1$ with validated hyperparameters.

which favors the use of BN layers in test-time adaptation and training [2], [3], [25], [26], could be interesting to reevaluate. In addition, the optimal stopping epoch is frequently not validated in the literature, and several studies only present test set performance over a limited number of iterations [4], [5], [28]. Figure 5 illustrates that performance does not consistently improve over time, underscoring the complexity of identifying the optimal number of iterations to perform.

Parameters to adapt. The choice of branch location is rarely justified in the literature. Some works plug in the branch in the middle of the core model [2], [5] or at the end of the feature encoder [3], [4], [6]. In semantic segmentation, the branch is sometimes placed at the beginning of the network [30], after the feature encoder [29], or at the end of the decoder [28]. Without a clear trend for the choice of k, we still observe that TTT along with BN layers works better for $k = 2, 3$, suggesting that adapting deeper layers for BN networks can be promising whereas $k = 1$ is a reasonable choice for IN. Regarding the parameters to adapt, some approaches focus solely on the affine parameters of the normalization layers, a common practice in test-time adaptation [26], [27]. In comparison, it is more common to adapt all parameters in test-time training [2], [3], [5], [28], [29]. We observe that training all parameters is generally a dangerous option since it can lead to performance degradation or model collapse while training only the affine parameters for both BN and IN layers is more stable and almost always leads to better results for IN.

Guidelines. To summarize our observations, we propose several guidelines to help with the implementation of TTT in practice. We further assess the final performances following our guidelines over the most severe domain shifts occurring in our datasets on Figure 4.

Guidelines for Test-Time Training in MS

1. Applying TTT is useful only when the model is facing a large domain shift/significant performance drop. In other cases, it can lead to non significant gains or even deterioration at the cost of additional computation time.

2. If the normalization of the core model can be chosen, using Instance Normalization layers is preferable to Batch Normalization if core model is adapted with TTT. Prefer BN otherwise.

3. Adapting only the affine parameters is the best choice in almost every case, leading to better and more stable results.

4. Adapting only shallower layers with Instance Normalization is preferable while adapting deeper layers with Batch Normalization can lead to better results.

5. On average, using Relative Patch Location (RPL) as the auxiliary task is a good choice, with consistent improvement for both normalization layers.

VI. Conclusion

MS lesion segmentation models tend to fail when the domain shift is pronounced. Nonetheless, TTT can significantly improve their performance even for single-patient adaptation. Our work offers a comprehensive study and provides insights into the practical challenges associated with applying TTT. We outline both the positive and negative aspects, along with the potential degradation that may arise from the misuse of these techniques. We hope that our work will contribute to a deeper investigation of the various aspects of TTT, particularly in its utilization in real-world scenarios.

Acknowledgment

This work was partly supported by the Walloon Region (Service Public de Wallonie Recherche, Belgium) under grant n°2010235 (ARIAC by DigitalWallonia.ai). The present research benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under the grant agreement n°1910247.

References

[1] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, "Dataset shift in machine learning," 2008.

[2] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *Int. Conf. Mach. Learn. (ICML)*. PMLR, 2020, pp. 9229–9248.

[3] Y. Liu, P. Kothari, B. Van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "Ttt++: When does self-supervised test-time training fail or thrive?" *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, pp. 21 808–21 820, 2021.

[4] Y. Gandelsman, Y. Sun, X. Chen, and A. Efros, "Test-time training with masked autoencoders," *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, pp. 29 374–29 385, 2022.

[5] D. Osowiechi, G. A. V. Hakim, M. Noori, M. Cheraghalikhani, I. Ben Ayed, and C. Desrosiers, "Tttflow: Unsupervised test-time training with normalizing flow," in *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2023, pp. 2126–2134.

[6] A. Bartler, A. Bühler, F. Wiewel, M. Döbler, and B. Yang, "Mt3: Meta test-time training for self-supervised test-time adaption," in *Int. Conf. Artif. Intell. Stat. (AIStat)*. PMLR, 2022, pp. 3080–3090.

[7] S. Valverde, M. Salem, M. Cabezas, D. Pareto, J. C. Vilanova, L. Ramió-Torrentà, À. Rovira, J. Salvi, A. Oliver, and X. Lladó, "One-shot domain adaptation in multiple sclerosis lesion segmentation using convolutional neural networks," *NeuroImage: Clinical*, vol. 21, p. 101638, 2019.

[8] A. Ackaouy, N. Courty, E. Vallée, O. Commowick, C. Barillot, and F. Galassi, "Unsupervised domain adaptation with optimal transport in multi-site segmentation of multiple sclerosis lesions from mri data," *Frontiers in computational neuroscience*, vol. 14, p. 19, 2020.

[9] S. Aslani, V. Murino, M. Dayan, R. Tam, D. Sona, and G. Hamarneh, "Scanner invariant multiple sclerosis lesion segmentation from mri," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2020, pp. 781–785.

[10] T. Varsavsky, M. Orbes-Arteaga, C. H. Sudre, M. S. Graham, P. Nachev, and M. J. Cardoso, "Test-time unsupervised domain adaptation," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part I 23*. Springer, 2020, pp. 428–436.

[11] R. A. Kamraoui, V.-T. Ta, T. Tourdias, B. Mansencal, J. V. Manjon, and P. Coupé, "Deeplesionbrain: Towards a broader deep-learning generalization for multiple sclerosis lesion segmentation," *Medical Image Analysis*, vol. 76, p. 102312, 2022.

[12] S. Cerri, O. Puonti, D. S. Meier, J. Wuerfel, M. Mühlau, H. R. Siebner, and K. Van Leemput, "A contrast-adaptive method for simultaneous whole-brain and lesion segmentation in multiple sclerosis," *Neuroimage*, vol. 225, p. 117471, 2021.

[13] C. Walton *et al.*, "Rising prevalence of multiple sclerosis worldwide: Insights from the atlas of ms," *Multiple Sclerosis Journal*, vol. 26, no. 14, pp. 1816–1821, 2020.

[14] A. J. Solomon, R. T. Naismith, and A. H. Cross, "Misdiagnosis of multiple sclerosis: Impact of the 2017 mcdonald criteria on clinical practice," *Neurology*, vol. 92, no. 1, pp. 26–33, 2019.

[15] V. Popescu *et al.*, "Brain atrophy and lesion load predict long term disability in multiple sclerosis," *J. Neurol. Neurosurg. Psychiatry (JNNP)*, vol. 84, pp. 1082–1091, 2013.

[16] C. Egger *et al.*, "Mri flair lesion segmentation in multiple sclerosis: Does automated segmentation hold up with manual annotation?" *NeuroImage: Clinical*, vol. 13, pp. 264–270, 2017.

[17] O. Commowick, B. Combès, F. Cervenansky, and M. Dojat, "Automatic methods for multiple sclerosis new lesions detection and segmentation," *Frontiers in Neuroscience*, vol. 17, p. 1176625, 2023.

[18] C. Zeng, L. Gu, Z. Liu, and S. Zhao, "Review of deep learning approaches for the segmentation of multiple sclerosis lesions on brain mri," *Frontiers in Neuroinformatics*, vol. 14, p. 610967, 2020.

[19] Y. Ma, C. Zhang, M. Cabezas, Y. Song, Z. Tang, D. Liu, W. Cai, M. Barnett, and C. Wang, "Multiple sclerosis lesion analysis in brain magnetic resonance images: techniques and clinical applications," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 6, pp. 2680–2692, 2022.

[20] E. A. AlBadawy, A. Saha, and M. A. Mazurowski, "Deep learning for segmentation of brain tumors: Impact of cross-institutional training and testing," *J Med Phys.*, vol. 45, no. 3, pp. 1150–1158, 2018.

[21] A. Malinin *et al.*, "Shifts 2.0: Extending the dataset of real distributional shifts," *arXiv preprint arXiv:2206.15407*, 2022.

[22] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Int. Conf. Mach. Learn. (ICML)*. PMLR, 2015, pp. 97–105.

[23] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Int. Conf. Mach. Learn. (ICML)*. PMLR, 2015, pp. 1180–1189.

[24] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *Int. Conf. Mach. Learn. (ICML)*. PMLR, 2020, pp. 6028–6039.

[25] Z. Nado, S. Padhy, D. Sculley, A. D'Amour, B. Lakshminarayanan, and J. Snoek, "Evaluating prediction-time batch normalization for robustness under covariate shift," *arXiv preprint arXiv:2006.10963*, 2020.

[26] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *Int. Conf. Learn. Represent. (ICLR)*, 2021.

[27] M. Bateson, H. Lombaert, and I. Ben Ayed, "Test-time adaptation with shape moments for image segmentation," in *Med. Image Comput. Comput. Assist. Interv. (MICCAI)*. Springer, 2022, pp. 736–745.

[28] F. Lyu, M. Ye, A. J. Ma, T. C.-F. Yip, G. L.-H. Wong, and P. C. Yuen, "Learning from synthetic ct images via test-time training for liver tumor segmentation," *IEEE Trans. Med. Imaging (TMI)*, vol. 41, no. 9, pp. 2510–2520, 2022.

[29] S. Fu, Y. Lu, Y. Wang, Y. Zhou, W. Shen, E. Fishman, and A. Yuille, "Domain adaptive relational reasoning for 3d multi-organ segmentation," in *Med. Image Comput. Comput. Assist. Interv. (MICCAI)*. Springer, 2020, pp. 656–666.

[30] N. Karani, E. Erdil, K. Chaitanya, and E. Konukoglu, "Test-time adaptable neural networks for robust medical image segmentation," *Med. Image Anal. (MedIA)*, vol. 68, p. 101907, 2021.

[31] A. Taleb *et al.*, "3d self-supervised methods for medical imaging," *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, pp. 18 158–18 172, 2020.

[32] Y. Tang *et al.*, "Self-supervised pre-training of swin transformers for 3d medical image analysis," in *IEEE/CVF Conf. Comput. Vis. Pattern Recogn. (CVPR)*, 2022, pp. 20 730–20 740.

[33] F. La Rosa *et al.*, "Multiple sclerosis cortical and wm lesion segmentation at 3t mri: a deep learning method based on flair and mp2rage," *NeuroImage: Clinical*, vol. 27, p. 102335, 2020.

[34] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.