Synchronizing Task Behavior: Aligning Multiple Tasks during Test-Time Training

Supplementary Material

A. Additional Experimental Details

Experimental Settings. In the training phase within the source domain, we utilize the Adam optimizer [3] with a polynomial decay for the learning rate. We set the learning rate to 2×10^{-5} and the weight decay to 1×10^{-6} for training the networks. The batch size is 8, and we perform 60,000 iterations for training. We set λ^{TP}=λ^{TBS}=1, and reduce λ^{TBS} to 0.01 at the test time. During test time, we adopt the SGD optimizer to ensure stable convergence with the TTT loss. The learning rate remains the same. During test time, we update the network for each batch of data for up to 40 steps in an online manner.

Table 1. Hyperparameters for experiments.

Hyperparameter	Value
⌞ Scheduler	Polynomial Decay
⌞ Minibatch size	8
⌞ Backbone	ResNet50 [2]
⌞ Learning rate	0.00002
⌞ Weight Decay	0.000001
Train Time Training	
⌞ Optimizer	Adam [3]
⌞ Number of iterations	60000
⌞ Learning rate	0.00002
⌞ Weight Decay	0.000001
Test Time Training	
⌞ Optimizer	SGD
⌞ Minibatch size	8
⌞ Number of steps	40

Metrics. For semantic segmentation, we utilize the mean Intersection over Union (mIoU) metric. The performance of surface normal prediction was measured by calculating the mean angle distances between the predicted output and the ground truth. To evaluate depth estimation and edge detection, we use the Root Mean Squared Error (RMSE).

Datasets. To implement TTT in semantic segmentation tasks on different datasets (Taskonomy ↔ NYUD-v2, Taskonomy ↔ PASCAL-Context), we find shared class labels in each of the two datasets. For Taskonomy ↔ NYUD-v2, we use 6 shared classes: `table, tv, toilet, sofa, potted plant, chair`. For Taskonomy ↔ PASCAL-Context, we use 7 class labels: `refrigerator, table, toilet, sofa, bed, sink, chair`. We use the split of train/test following the common multi-task benchmarks, NYUD-v2, PASCAL-Context and Taskonomy. In the case of NYUD-v2, we utilize 795 images for training and reserve 654 images for test-time training. With PASCAL-Context, 4,998 images are employed during training, and 5,105 images are used for test-time training. For Taskonomy, we leverage 295,521 images for training and apply 5,451 images during test-time.

B. Evaluation of Synchronization

We qualify the limitations of the existing TTT methods on multi-task settings through three points: step variance (SV) of peak step during adaptation, dynamic time warping (DTW) and cosine similarity (CS) of adaptation graph. We denote that the steps $\tau \in \{1, 2, \ldots, \mathrm{T}\}$ and $y_i^{(\tau)}$ denotes the performance of i-th task at step τ. n is the number of tasks. Step Variance measures the dispersion of the steps where tasks achieve their maximum performance:

$$\mathrm{SV} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\tau_i^{peak} - \bar{\tau}^{peak}\right)^2}. \tag{1}$$

where τ_i^{peak} denotes the step with best performance for each task i and $\bar{\tau}^{peak} = \frac{1}{n}\sum_{i=1}^{n}\tau_i^{peak}$.

DTW aligns the performance trajectories Y_i and Y_j of the different tasks by computing the optimal alignment path:

$$\mathrm{DTW} = \frac{1}{C(n,r)}\sum_{i=1}^{n}\sum_{j>i}^{n}\mathrm{DC}_{i,j}(T,T),$$
$$\mathrm{DC}_{i,j}(\tau,\kappa) = d(y_i^{(\tau)}, y_j^{(\kappa)}) + \min\begin{cases}\mathrm{DC}_{i,j}(\tau-1,\kappa),\\ \mathrm{DC}_{i,j}(\tau,\kappa-1),\\ \mathrm{DC}_{i,j}(\tau-1,\kappa-1)\end{cases} \tag{2}$$

where d represents a distance metric, with the Euclidean distance used in our case, and C denotes the binomial coefficient, representing the number of combinations.

Cosine Similarity evaluates the directional consistency of performance changes between two tasks:

$$\text{Cosine Similarity} = \frac{1}{C(n,r)}\sum_{i=1}^{n}\sum_{j>i}^{n}\sum_{\tau=1}^{T-1}\mathrm{CS}_{i,j}(\tau), \tag{3}$$

$$\mathrm{CS}_{i,j}(\tau) = \frac{\Delta y_i^{\tau}\cdot\Delta y_j^{\tau}}{\|\Delta y_i^{\tau}\|\|\Delta y_j^{\tau}\|}, \tag{4}$$

where $\Delta y_i^{\tau} = y_i^{\tau+1} - y_i^{\tau}$, and C represents the binomial coefficient.

C. Reliability of Assumptions

Reliability of Assumption 1. Assumption 1 posits that task relations in the target domain can be reliably captured by learning a domain-dependent transformation function f during test-time training. To support this, we compute task affinity matrices using the cosine similarity between task-specific latents learned in the source domain. We compare these with the affinity matrices obtained from the fine-tuned model on the target domain, plotting the gap in Fig. 1, which shows the domain shift from Taskonomy to NYUD-v2. While the initial task relations differ across domains (*Before Adaptation*), the learned transformation f effectively reduces the affinity gap (*After Adaptation*), supporting the validity of the assumption in real-world scenarios.

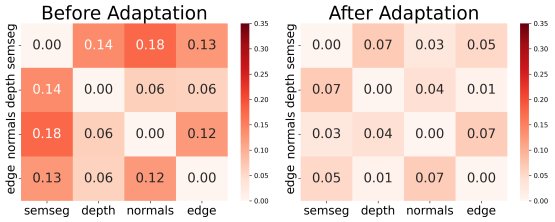

Figure 1. Task affinity gap before and after adaptation.

Reliability of Assumption 2. Assumption 2 does not claim that TBS achieves perfect predictions of task labels, but rather that its predictions from masked latents $\tilde{z}$ are *comparable* to those from unmasked latents z once sufficiently trained. We evaluate this by directly measuring the TBS prediction loss $\mathcal{L}^{TBS}$ under varying mask ratios. As shown in Fig. 2, TBS maintains over 90% of its original performance at $r = 0$ even when the mask ratio increases to $r = 0.9$, supporting the validity of the assumption in practice. This assumption is not directly comparable to those in image-level restoration settings like MAE, as TBS predicts task labels from masked latent features, rather than reconstructing pixel-level content.

Additionally, to quantify the level of distribution shift under which our assumptions hold, we evaluate $\triangle_{TTT}$ by adding Gaussian noise to input images, where the noise standard deviation is set to $\alpha \cdot \sigma_{\text{img}}$ (α: noise scale), with σ_{img} being the standard deviation of input images. As shown in Tab. 2, the results suggest that our method and proposition hold robustly under moderate shifts.

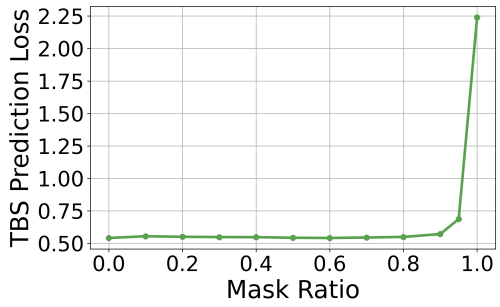

Figure 2. TBS prediction loss across varying masking ratios.

Table 2. $\triangle_{TTT}$ under Gaussian noise w.r.t. noise scale α

α	0	0.01	0.05	0.1	0.2	0.3	0.4
$\triangle_{TTT} \uparrow$	34.89	34.42	33.80	32.15	28.64	23.80	19.73

D. Additional Experiments and Analyses

Comparison with Previous Methods in Different Scenarios. We compare S4T with previous state-of-the-art TTT methods in different scenarios, using NYUD-v2 and PASCAL-Context as the source domains and Taskonomy as the target domain. The results are presented in Tab. 4. For a fair comparison, we select the point at which each method achieves its best TTT performance, averaged across all tasks, as measured by $\triangle_{TTT}$. Since NYUD-v2 and PASCAL-Context have smaller datasets, the overall TTT performance is lower compared to scenarios where Taskonomy is used as the source domain. The proposed S4T still demonstrates comparable performance in these scenarios.

Evaluation of S4T Using Single Task for Adaptation. To ensure that S4T's gains are not solely due to multi-task structures, we evaluated it in a single-task setting where only one task label is available during adaptation. In this case, TBS uses a single latent vector per task. As shown in Tab. 5, performance drops compared to the multi-task setup, highlighting the importance of modeling task relations. Still, S4T achieves competitive results, confirming that its benefits stem from the core TTT mechanism rather than architectural scale or multi-head design.

Number of Parameters and Inference Time. As shown in Table 6, we compare the number of parameters and inference time from Taskonomy to NYUD-v2 for S4T, against previous SOTA TTT methods. For inference, we use NVIDIA RTX A6000 48GB with batch size 8.

Comparison with a Strong MTL+TTT Baseline. We compare S4T with a strong TTT baseline that combines an existing TTT method (NC-TTT [9]) and a high-performing multi-task learning MTL architecture (MTI-Net [10]). As shown in Tab. 3, S4T achieves slightly higher adaptation gain ($\triangle_{\text{TTT}} = 34.9$ vs. 32.5) while using fewer than 20% of the parameters (29.2M vs. 165.0M) and maintaining comparable inference time (0.3657s vs. 0.3208s). This indicates that our method is not only effective in terms of performance but also significantly more parameter-efficient. Importantly, S4T is designed as a modular plug-in that can be ad-hoc attached to existing MTL architectures.

Table 3. Comparison of S4T and NC-TTT + MTI-Net

Method	$\triangle_{TTT} \uparrow$	Params (M)	Inference Time (s)
NC-TTT + MTI-Net	32.5	165.0	0.3208
Ours	34.9	29.2	0.3657

Table 4. Comparison of multi-task performance from NYUD-v2 to Taskonomy across different tasks for S4T, against previous TTA and TTT methods.

Dataset	NYUD-v2 → Taskonomy					PASCAL-Context → Taskonomy			
Task	Semseg	Depth	Normal	Edge	$\triangle_{TTT}$	Semseg	Normal	Edge	$\triangle_{TTT}$
Metric	mIoU ↑	RMSE ↓	mErr ↓	RMSE ↓	% ↑	mIoU ↑	mErr ↓	RMSE ↓	% ↑
Base	48.21 ±1.580e-2	0.0507 ±2.000e-4	27.60 ±1.200e-2	0.3058 ±2.000e-4	+0.00	50.94 ±6.630e-1	31.27 ±7.100e-2	0.3032 ±1.000e-4	0.00
TENT [11]	39.75 ±1.200e-2	0.0634 ±0.000e+0	37.49 ±3.500e-1	0.3084 ±5.000e-4	-19.76	44.68 ±3.530e-1	42.32 ±1.830e-1	0.3269 ±2.100e-4	-0.18
TIPI [6]	47.03 ±7.080e-4	0.0514 ±3.550e-8	28.40 ±1.490e-5	0.3052 ±3.320e-8	-1.64	51.48 ±1.200e-3	32.33 ±4.180e-5	0.3031 ±1.490e-7	-0.77
TTT [9]	49.66 ±4.120e-1	0.0523 ±4.100e-3	31.96 ±1.190e-1	0.3094 ±6.000e-4	-4.28	48.00 ±2.661e+0	37.77 ±2.214e+0	0.3048 ±3.600e-4	-9.00
TTT++ [4]	39.29 ±1.089e-1	0.0595 ±1.300e-3	36.53 ±4.160e-1	0.3132 ±2.610e-3	-17.65	38.66 ±3.090e-1	39.81 ±3.850e-1	0.3050 ±6.480e-4	-17.33
TTTFlow [7]	48.56 ±3.000e-1	0.0540 ±1.000e-4	34.36 ±1.425e-1	0.3086 ±1.000e-4	-7.73	51.55 ±3.950e-1	34.60 ±1.200e-1	0.3042 ±1.500e-3	-3.26
ClusT3 [1]	51.14 ±1.757e+0	0.0516 ±3.700e-4	30.03 ±4.310e-1	0.3065 ±5.000e-4	-1.16	49.67 ±6.480e-1	35.22 ±1.570e-1	0.3019 ±1.900e-4	-4.90
ActMAD [5]	55.04 ±7.300e-4	0.0506 ±5.800e-8	27.88 ±7.000e-5	0.3081 ±1.100e-9	+3.17	51.79 ±8.030e-1	31.10 ±1.240e-1	0.3031 ±1.020e-4	+0.74
NC-TTT [8]	49.95 ±6.530e-1	0.0516 ±4.600e-5	29.95 ±4.200e-2	0.3093 ±1.010e-4	-1.96	48.78 ±5.100e-1	32.86 ±2.220e+0	0.3040 ±1.300e-3	-3.19
S4T (ours)	53.12 ±1.340e-1	0.0511 ±1.930e-4	27.58 ±1.044e-1	0.3089 ±3.540e-5	**+2.13**	53.18 ±3.150e-1	31.50 ±7.890e-2	0.3036 ±2.000e-4	**+1.18**

Table 5. We compare the TTT performance of Taskonomy as the source domain and NYUD-v2 as the target domain across four tasks for S4T, analyzing both single-task and multi-task scenarios.

Dataset	NYUD-v2 → Taskonomy					PASCAL-Context → Taskonomy			
Task	Semseg	Depth	Normal	Edge	$\triangle_{TTT}$	Semseg	Normal	Edge	$\triangle_{TTT}$
Metric	mIoU ↑	RMSE ↓	mErr ↓	RMSE ↓	% ↑	mIoU ↑	mErr ↓	RMSE ↓	% ↑
Base	29.31 ±6.300e-2	1.179 ±8.000e-3	61.32 ±8.200e-1	0.1443 ±7.100e-4	+0.00	27.08 ±1.400e-2	63.46 ±9.540e-1	0.1185 ±7.100e-4	0.00
S4T (single)	56.56 ±8.500e-2	1.065 ±5.600e-3	47.32 ±3.000e-2	0.1444 ±1.900e-5	-	43.28 ±3.700e-1	46.91 ±7.500e-2	0.1185 ±9.600e-6	-
S4T (multi)	59.37 ±1.520e-1	1.052 ±7.500e-3	45.33 ±7.200e-2	0.1441 ±5.100e-5	+34.94	45.42 ±1.900e-1	41.41 ±6.300e-1	0.1183 ±5.000e-5	+34.20

Table 6. Comparisons of number of parameters and inference time from Taskonomy to NYUD-v2 for S4T, against previous TTT methods.

	TTT	TTT++	TTTFlow	ClusT3	ActMad	NC-TTT	S4T(Ours)
Number of Parameters	27.49M	149.40M	49.56M	23.59M	23.55M	23.91M	29.24M
Inference Time (s)	0.4016	2.7291	0.4355	0.2459	0.3041	0.1319	0.3657

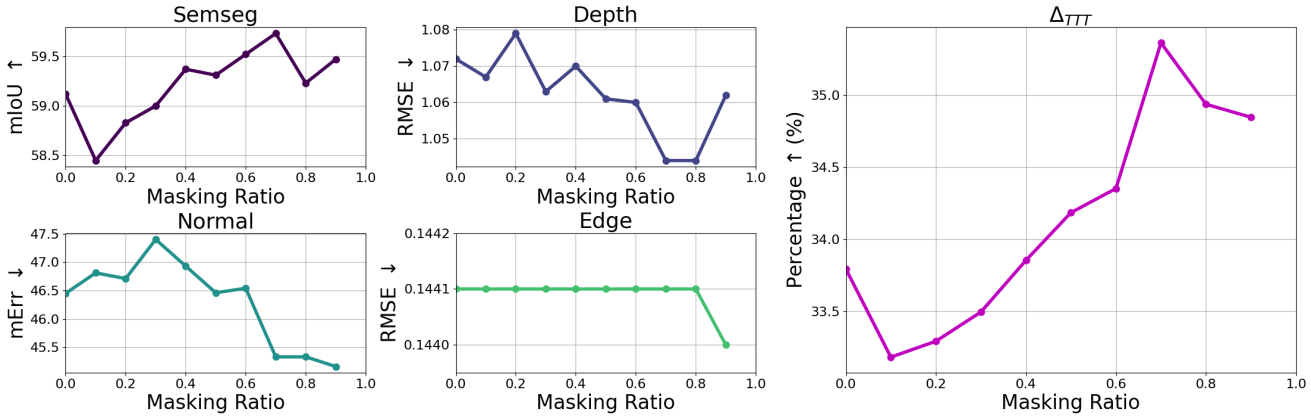

Figure 3. Ablation study on the masking ratio of S4T. We evaluate performance under the domain shift from Taskonomy to NYUD-v2.

E. Derivations of ??

For simplicity, denote the task-specific latent space as $\{z_{t,i}\}_{i=1}^n$ and its masked version as $\{\tilde{z}_{t,i}\}_{i=1}^n$.

$$
\begin{aligned}
& d(\theta, p(\{z_{t,i}\}_{i=1}^n, y_{t,j})) - d(\theta, p(\{\tilde{z}_{t,i}\}_{i=1}^n, y_{t,j})) && (5)\\
=& \mathbb{E}_{p(\{z_{t,i}\}_{i=1}^n)}[d[p(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{z_{t,i}\}_{i=1}^n)]] && (6)\\
& - \mathbb{E}_{p(\{\tilde{z}_{t,i}\}_{i=1}^n)}[d[p(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (7)\\
=& \mathbb{E}_{p(\{z_{t,i}\}_{i=1}^n, \{\tilde{z}_{t,i}\}_{i=1}^n)}[d[p(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{z_{t,i}\}_{i=1}^n)]] && (8)\\
& - \mathbb{E}_{p(\{z_{t,i}\}_{i=1}^n, \{z_{t,i}\}_{i=1}^n \{\tilde{z}_{t,i}\}_{i=1}^n)}[d[p(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (9)\\
\leq& \mathbb{E}_{p(\{z_{t,i}\}_{i=1}^n, \{\tilde{z}_{t,i}\}_{i=1}^n)}[d[p_\theta(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (10)\\
& + \mathbb{E}_{p(\{z_{t,i}\}_{i=1}^n, \{\tilde{z}_{t,i}\}_{i=1}^n)}[d[p(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (11)
\end{aligned}
$$

The Eq. (11) follows from the triangle inequality.
Rearranging the above equation results in the following inequality.:

$$
\begin{aligned}
d(\theta, p(\{z_{t,i}\}_{i=1}^n, y_{t,j})) \leq & \; d(\theta, p(\{\tilde{z}_{t,i}\}_{i=1}^n, y_{t,j})) && (12)\\
& + \mathbb{E}_{p(\{z_{t,i}\}_{i=1}^n, \{\tilde{z}_{t,i}\}_{i=1}^n)}[d[p_\theta(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (13)\\
& + \mathbb{E}_{p(\{z_{t,i}\}_{i=1}^n, \{\tilde{z}_{t,i}\}_{i=1}^n)}[d[p(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (14)
\end{aligned}
$$

In the multi-task setting, we apply Eq. (14) to each task as follows:

$$
\begin{aligned}
\sum_{j=1}^n d(\theta, p(\{z_{t,i}\}_{i=1}^n, y_{t,j})) \leq & \sum_{j=1}^n d(\theta, p(\{\tilde{z}_{t,i}\}_{i=1}^n, y_{t,j})) && (15)\\
& + \sum_{j=1}^n \mathbb{E}_{p(\{z_{t,i}, \tilde{z}_{t,i}\}_{i=1}^n)}[d[p_\theta(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (16)\\
& + \sum_{j=1}^n \mathbb{E}_{p(\{z_{t,i}, \tilde{z}_{t,i}\}_{i=1}^n)}[d[p(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (17)\\
\leq & \sum_{j=1}^n d(\theta, p(\{\tilde{z}_{t,i}\}_{i=1}^n, y_{t,j})) && (18)\\
& + \sum_{j=1}^n \mathbb{E}_{p(\{z_{t,i}, \tilde{z}_{t,i}\}_{i=1}^n)}[d[p_\theta(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (19)\\
& + C \cdot \sum_{j=1}^n \mathbb{E}_{p(\{z_{s,i}, \tilde{z}_{s,i}\}_{i=1}^n)}[d[p(y_{s,j}|\{z_{s,i}\}_{i=1}^n), p(y_{s,j}|\{\tilde{z}_{s,i}\}_{i=1}^n)]] && (20)
\end{aligned}
$$

The inequality between Eq. (17) and Eq. (20) holds under Assumption **??** with a scaling factor C, which asserts that task relations remain proportionally consistent between tasks and their masked counterparts. With a properly chosen masking ratio, TBS effectively captures task relations in the source domain, as ensured by Assumption **??**. Consequently, Eq. (20) approaches zero, aligning with the training objective in the source domain. Therefore, the following inequality holds:

$$
\begin{aligned}
\sum_{j=1}^n d(\theta, p(\{z_{t,i}\}_{i=1}^n, y_{t,j})) \leq & \sum_{j=1}^n d(\theta, p(\{\tilde{z}_{t,i}\}_{i=1}^n, y_{t,j})) && (21)\\
& + \sum_{j=1}^n \mathbb{E}_{p(\{z_{t,i}, \tilde{z}_{t,i}\}_{i=1}^n)}[d[p_\theta(y_{t,j}|\{z_{t,i}\}_{i=1}^n), p_\theta(y_{t,j}|\{\tilde{z}_{t,i}\}_{i=1}^n)]] && (22)
\end{aligned}
$$

References

[1] Gustavo A Vargas Hakim, David Osowiechi, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Ismail Ben Ayed, and Christian Desrosiers. Clust3: Information invariant test-time training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6136–6145, 2023. 3

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[4] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34: 21808–21820, 2021. 3

[5] Muhammad Jehanzeb Mirza, Pol Jané Soneira, Wei Lin, Mateusz Kozinski, Horst Possegger, and Horst Bischof. Actmad: Activation matching to align distributions for test-time-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24152–24161, 2023. 3

[6] A Tuan Nguyen, Thanh Nguyen-Tang, Ser-Nam Lim, and Philip HS Torr. Tipi: Test time adaptation with transformation invariance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24162–24171, 2023. 3

[7] David Osowiechi, Gustavo A Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ismail Ben Ayed, and Christian Desrosiers. Tttflow: Unsupervised test-time training with normalizing flow. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2126–2134, 2023. 3

[8] David Osowiechi, Gustavo A Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ali Bahri, Moslem Yazdanpanah, Ismail Ben Ayed, and Christian Desrosiers. Nc-ttt: A noise constrastive approach for test-time training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6078–6086, 2024. 3

[9] 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*, pages 9229–9248. PMLR, 2020. 2, 3

[10] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 527–543. Springer, 2020. 2

[11] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 3