TTT-KD: Test-Time Training for 3D Semantic Segmentation through Knowledge Distillation from Foundation Models

Supplementary Material

A. Limitations

Despite the number of benefits of our TTT-KD, it is not totally exempt from limitations. Similar to other Test-Time Training (TTT) methods, the main limitation is the additional computation required for TTT in comparison to simply evaluating a network with frozen weights. Our *standard* TTT-KD requires processing each image with a foundation model and then performing several optimization steps for each scene. Still, this could be drastically reduced with our online version, which only optimizes the network weights for a single step, as other Test-Time Adaptation (TTA) methods, and shows considerable improvements even surpassing the offline version for some datasets. This additional cost is further reduced if we use a stride between parameters updates or momentum in the SDG optimizer (see ablation studies).

Another limitation is that our algorithm relies on the robustness of the foundation model to out-of-distribution (OOD) data. From the experiments presented in the paper, and additional experiments provided in the supplementary material, we concluded that DINOv2 [12] is robust to domain shifts. However, we acknowledge that there might be some cases where the foundation model used does not present such robustness. In such cases, existing TTT or TTA approaches for 2D images could be used to adapt the foundation model to this new domain.

B. Ablation studies

In this section, we describe the ablation studies carried out to investigate the effect of different design choices on the performance of our algorithm. Unless otherwise stated, due to computational reasons, all these ablations are performed on the task of indoor 3D semantic segmentation while adapting from S3DIS $\rightarrow$ SCANNET , with 25 TTT steps using our offline setup.

B.1. Number of TTT steps

We measure the performance of our TTT-KD algorithm in relationship to the number of TTT steps and plot the results in Fig. 1 (a). We see that the mean Intersection over Union (mIoU) increases with the number of TTT updates, and saturates at 200 steps. However, relative improvement is reduced after 100 steps.

B.2. Number of images

Our algorithm relies on paired pointcloud and image data, which might be restrictive for some setups. Therefore, we measure the performance of our method w.r.t. the number of images used for Knowledge Distillation (KD). Fig. 1 (b) presents the results of this experiment. We can see that even when only a single image is used (most of the points in the scene do not have a paired image), we can achieve a boost in mIoU of 4.5. These results support the findings on the outdoor tasks, where also only one image is available for adaptation. Moreover, we can see that the improvement saturates for 5 images when the improvement obtained by including an additional image is reduced.

B.3. Foundation model

For a more generalizable TTT approach, it must work with any of the readily available *off-the-shelf* foundation models. To evaluate this, we experiment with different foundation

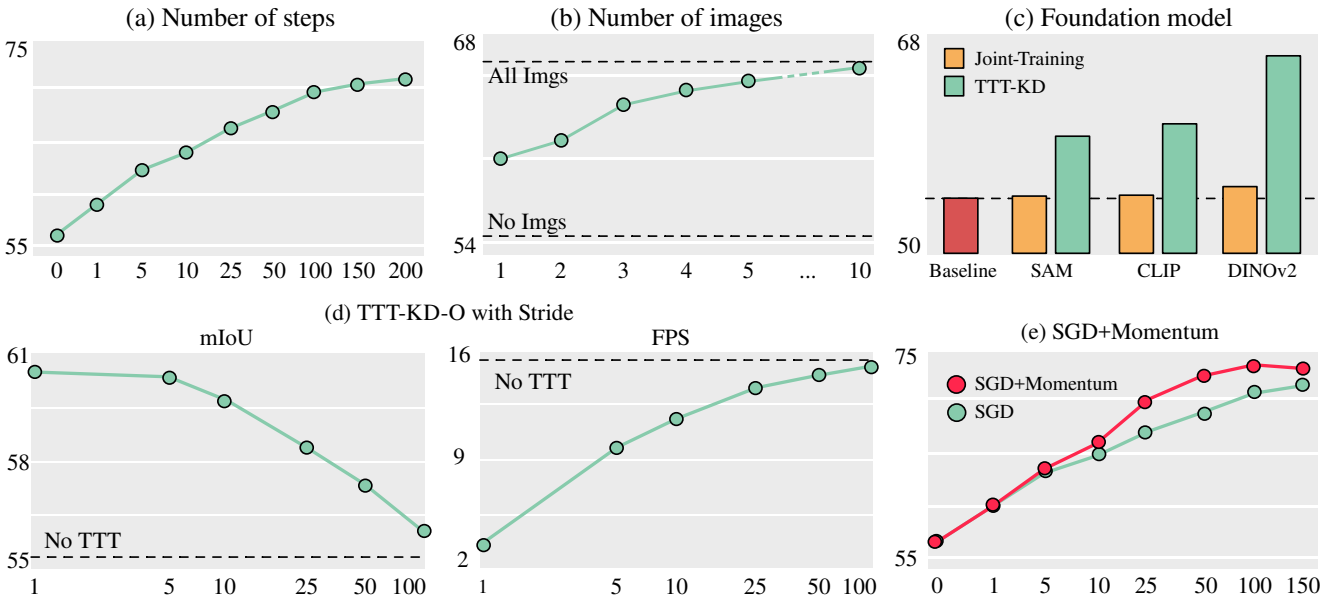

Figure 1. **Ablation studies. (a)** Improvement w.r.t. the number of TTT step utilized per scene. **(b)** mIoU of the model w.r.t. to the number of images used in our TTT-KD . **(c)** Comparison of DINOv2 [12] to CLIP [14] and SAM [8] as our foundation model. **(d)** mIoU and FPS when updating the model's parameters using our TTT-KD-O algorithm with stride, *i.e.* only performing a TTT step every s predictions. **(e)** Performance of the model optimized using SGD vs SGD+Momentum.

models used for TTT-KD. In this experiment, we compare the foundation model used in our main experiments, DINOv2 [12], to a CLIP [14] model trained with 2 B paired image-text data, and to a SAM [8] model trained with 11 M annotated images, and provide the results in Fig. 1 (c). Although previous works [5, 12] have shown that DINOv2 provides better segmentation masks than CLIP, our TTT-KD is also able to provide considerable performance gains while using CLIP in our pipeline. Moreover, TTT-KD is able to obtain similar results when the foundation model is trained with ground truth image segmentation masks, such as the SAM model.

B.4. Stride in TTT-KD-O

In this section, we analyze how the frequency of parameter updates in our online TTT setup affects the performance. Fig. 1 (d) shows, for the S3DIS $\rightarrow$ ScanNet setup, the mIoU for a different number of predictions in between each parameter update, *i.e.* a stride of 5 indicates that the parameters of the model are updated every 5 scene predictions. We can see that even if our TTT-KD-O is sparsely applied, this still leads to a significant improvement while reducing the computational burden and being able to maintain almost 10 or more predictions per second.

B.5. Momentum in SGD

Fig. 1 (e) presents the results of optimizing the parameters of the model in our TTT-KD setup using SGD vs. SGD+Momentum with a momentum equal to 0.9. The results show that momentum converges faster, but with too many iterations the performance starts degrading, making SGD+Momentum ideal for a setup that requires a limited number of TTT iterations.

C. Computational cost

TTT-KD is computationally very inexpensive. All our experiments are performed on a single NVIDIA $A6000$ GPU, requiring 2 Gb memory and, on average, 177 ms for each TTT step in the ScanNet dataset, plus 210 ms for each image processed by the foundation model, DINOv2 ViT-L. However, if speed is a concern for some applications, we can reduce the frequency of parameter updates in our online TTT setup as shown in Fig. 1 (d), being able to maintain almost 10 or more predictions per second. Additionally, since the main cost of our algorithm is the image processing with the foundation model, we could use a smaller model, such as the distilled ViT-S version of DINOv2, reducing the time from 210 ms per image to 25 ms.

D. ScanNet benchmark

Since our TTT-KD provides a clear improvement on in-distribution (ID) data, we compare our strategy to SOTA models on the ScanNet benchmark for 3D semantic segmentation. For this experiment, we decrease the resolution of the subsampling step to 2 cm and keep the same data aug-

Table 1. Results on the test and validation sets of SCANNET .

Method	Res.	Val.	Test
MinkowskiNet [3]	2cm	72.2	73.6
Point Transf. V2 [19]	2cm	75.4	75.2
PNE [7]	2cm	74.9	75.5
OctFormer [18]	1cm	75.7	76.6
Point Transf. V3 [19]	2cm	<u>77.5</u>	**77.9**
Joint-Train	2cm	75.7	–
TTT-KD		**77.6**	<u>77.3</u>

mentation techniques as in our main experiments. Table 1 presents the result of this experiment. We can see that our model trained jointly with the self-supervised task provides a competitive validation accuracy over recent supervised methods trained solely on SCANNET . However, when we perform TTT with our TTT-KD algorithm, we can see that it outperforms other methods on the validation set while achieving competitive performance on the test set. Please note that historically TTT methods have been proposed to adapt to OOD data only [6, 10, 16]. However, our TTT-KD also shows impressive performance gains while adapting to ID data.

E. TTT-KD algorithm

In Alg. 1 we present the detailed TTT-KD algorithm, consisting of the joint-training, the TTT, and inference phases.

F. Additional qualitative results

Fig. 2 provides additional qualitative results.

G. Datasets

G.1. Data augmentations

Although it is common practice to use specific data augmentations for each data set, this might lead to a bigger generalization gap when evaluating on OOD datasets. Therefore, for a fair baseline, we use a fixed set of data augmentations for all the data sets. We use random mirror of the X, Y axes, random rotations around the up vector, random scaling, elastic distortion, jitter of point coordinates, random crop, random translation, random adjustments of brightness and contrast of the point's colors, and RGB shift. Moreover, we subsample the scene using a voxel size of $4\,$cm. Lastly, we mix two scenes [11] with a probability equal to 0.5.

G.2. ScanNet [4]

This dataset is composed of $1,513$ real 3D scans of different rooms, for which each point in the scan is classified among 20 different classes. Moreover, for each scan, a set of images used for the 3D reconstruction is provided. The dataset is divided into two splits, $1,201$ rooms for training and 312 rooms for validation. Moreover, data samples from an additional 100 rooms are provided as a test set for benchmarking where ground truth labels are not available. In our experiments, we train the models on the train set and report performance on the validation set.

G.3. S3DIS [1]

This dataset provides dense pointclouds of 271 different rooms from 6 large-scale areas. Each point in the dataset is classified among 13 different classes. The dataset also provides the 2D images used for the 3D reconstruction. Following previous works [17, 21], we use data collected in areas 1, 2, 3, 4, and 6 for training and data collected in area 5 for testing.

G.4. Matterport3D [2]

Matterport3D provides pointclouds from 90 building-scale scenes, each composed of multiple regions or areas, and a set of images used for the 3D reconstruction. Each point within the dataset is classified among 21 classes. We follow the official splits and use 61 scenes for training, 11 for validation, and 18 for testing. We report the performance of the models on the test split.

H. Semantic labels

Tbl. 2 presents the list of classes per each dataset. We can see that SCANNET and MATTERPORT3D share almost all classes with the exception of *ceilling*. However, with S3DIS the number of shared classes is only 8 with SCANNET and 9 with MATTERPORT3D .

I. Baselines

To implement the baselines compared in our main experiments, we followed the original implementations available for all the methods. However, TTT-MAE [10], does not provide an architecture or setup for the task of 3D semantic segmentation of scenes composed of a variable number of points. Moreover, since the original TTT-MAE was originally designed for the PointMAE [13] architecture, it is not easily plugged into other network architectures used for the task of 3D semantic segmentation as the Minkowski34C [3] model. Therefore, in order to be consistent in our experiments and use the same network architectures in all experiments, we implemented the TTT-MAE baseline by using an MAE self-supervised task specifically designed for 3D scene understanding [20].

Table 2. List of all semantic labels used in our experiments and their presence in each individual dataset.

	Bathtub	Beam	Bed	Board	Bookshelf	Cabinet	Ceiling	Chair	Clutter	Column	Counter	Curtain	Desk	Door	Floor	Otherfurniture	Picture	Refrigerator	Shower curtain	Sink	Sofa	Table	Toilet	Wall	Window
S3DIS		✓		✓	✓		✓	✓	✓	✓				✓	✓						✓	✓		✓	✓
SCANNET	✓		✓		✓	✓		✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MATTERPORT3D	✓		✓		✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

J. Training details

J.1. Training

We use AdamW [9] as our optimizer with a maximum learning rate of 0.005 and a OneCycleLR learning rate scheduler [15] with an initial division factor of 10, and a final factor of 1000. To prevent overfitting, we use a weight decay value of 0.0001 and label smoothing for the segmentation task with a value of 0.2.

J.2. Test-time training

For TTT, we use SGD without momentum and a learning rate equal to 1.0. We optimize 100 steps for each scene before performing the final prediction.

J.3. Number of images

For SCANNET , we used the images provided in the 25 K subset, where images from the RGB-D video sequence are taken at intervals of 100. For MATTERPORT3D and S3DIS we select the available images for each scene from the provided 2D data. If more than 50 images for each scene are available, we randomly select 50.

Algorithm 1: TTT-KD algorithm.

Input: Training data $\mathcal{S} = \{(\mathcal{X}, \mathcal{F}, \mathcal{Y}, \mathcal{I}, \mathcal{U})\}$
Test sample $(\mathcal{X}', \mathcal{F}', \mathcal{I}', \mathcal{U}')$
3D Backbone (ψ_{3D})
2D Foundation Model (ϕ_{2D})
Projectors $(\rho_{\mathcal{Y}}, \rho_{2D})$
Result: Prediction $(\mathcal{Y}')$
begin
  # Joint-Training
  for *numEpochs* **do**
    $(\mathcal{X}, \mathcal{F}, \mathcal{Y}, \mathcal{I}, \mathcal{U}) = \text{sampleBatch}(\mathcal{S})$
    $F^{3D} = \psi_{3D}(\mathcal{X}, \mathcal{F})$
    $\hat{\mathcal{Y}} = \rho_{\mathcal{Y}}(F^{3D})$
    $\hat{F}^{2D} = \rho_{2D}(F^{3D})$
    $F^{2D} = \phi_{2D}(\mathcal{I})$
    $\mathcal{L} = \mathcal{L}_{\mathcal{Y}}(\hat{\mathcal{Y}}, \mathcal{Y}) + \mathcal{L}_{2D}(\hat{F}^{2D}, F^{2D}, \mathcal{U})$
    $\mathcal{L}$.backward()
    optimizer.step()
  $\mathcal{Y}' = 0$
  for *numRotations* **do**
    $\hat{\mathcal{X}}' = \text{Rotate}(\mathcal{X}')$
    # Test-Time Training
    for *numTTTSteps* **do**
      $\hat{F}^{2D} = \rho_{2D}(\psi_{3D}(\hat{\mathcal{X}}', \mathcal{F}'))$
      $F^{2D} = \phi_{2D}(\mathcal{I}')$
      $\mathcal{L} = \mathcal{L}_{2D}(\hat{F}^{2D}, F^{2D}, \mathcal{U}')$
      $\mathcal{L}$.backward()
      optimizer.step()
    # Inference
    $\mathcal{Y}' \mathrel{+}= \rho_{\mathcal{Y}}(\psi_{3D}(\mathcal{X}', \mathcal{F}'))$

References

[1] I. Armeni, S. Sax, A. R Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding, 2017. 1, 3

[2] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 1, 3

[3] C. Choy, J. Y. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3, 5

[4] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *Proc. CVPR*, 2017. 1, 3

[5] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2023. 2

[6] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022. 3

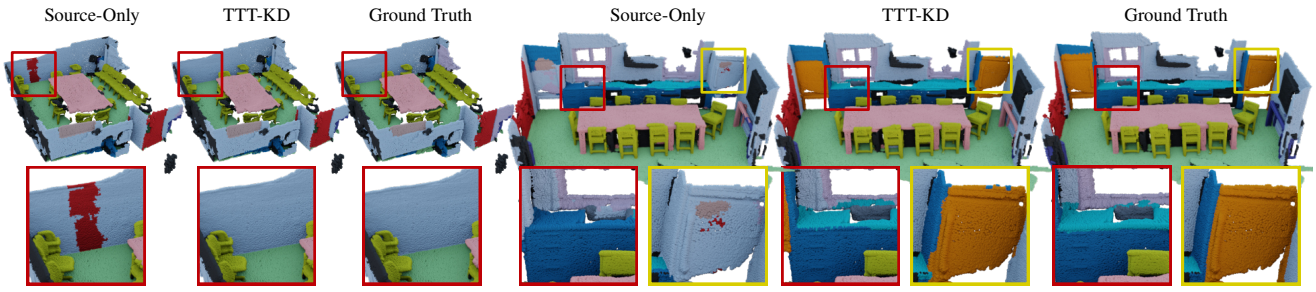

Figure 2. Additional qualitative results of the Mink [3] network trained and tested on the SCANNET dataset. We can see that even if training and testing data are from the same distribution, our TTT-KD algorithm also improves the predictions of the model.

[7] P. Hermosilla. Point neighborhood embeddings, 2023. 3

[8] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 2

[9] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proc. ICLR*, 2019. 4

[10] M. J. Mirza, I. Shin, W. Lin, A. Schriebl, K. Sun, J. Choe, M. Kozinski, H. Possegger, I. S. Kweon, K.-J. Yoon, and H. Bischof. Mate: Masked autoencoders are online 3d test-time learners. *Proc. ICCV*, 2023. 3

[11] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In *International Conference on 3D Vision (3DV)*, 2021. 3

[12] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1, 2

[13] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022. 3

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning (ICML)*, 2021. 2

[15] L. N. Smith and N. Topin. Super-convergence: Very fast training of residual networks using large learning rates, 2017. 4

[16] 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. 3

[17] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 3

[18] Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. *ACM Transactions on Graphics (SIGGRAPH)*, 2023. 3

[19] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[20] Xiaoyang Wu, Xin Wen, Xihui Liu, and Hengshuang Zhao. Masked scene contrast: A scalable framework for unsupervised 3d representation learning. In *IEEE / CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[21] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 3