MATE: Masked Autoencoders are Online 3D Test-Time Learners

M. Jehanzeb Mirza[†1,2] Inkyu Shin[†3] Wei Lin[†1] Andreas Schriebl[1] Kunyang Sun[4] Jaesung Choe[3]
Mateusz Kozinski[1] Horst Possegger[1] In So Kweon[3] Kuk-Jin Yoon[3] Horst Bischof[1,2]

[1]Institute for Computer Graphics and Vision, Graz University of Technology, Austria
[2]Christian Doppler Laboratory for Embedded Machine Learning
[3]Korea Advanced Institute of Science and Technology (KAIST), South Korea [4]Southeast University, China

Abstract

Our MATE is the first Test-Time-Training (TTT) method designed for 3D data, which makes deep networks trained for point cloud classification robust to distribution shifts occurring in test data. Like existing TTT methods from the 2D image domain, MATE also leverages test data for adaptation. Its test-time objective is that of a Masked Autoencoder: a large portion of each test point cloud is removed before it is fed to the network, tasked with reconstructing the full point cloud. Once the network is updated, it is used to classify the point cloud. We test MATE on several 3D object classification datasets and show that it significantly improves robustness of deep networks to several types of corruptions commonly occurring in 3D point clouds. We show that MATE is very efficient in terms of the fraction of points it needs for the adaptation. It can effectively adapt given as few as 5% of tokens of each test sample, making it extremely lightweight.

1. Introduction

Recent deep neural networks show impressive performance in classifying 3D point clouds. However, their success is warranted only if the test data originates from the same distribution as training data. In real-world scenarios, this assumption is often violated. A LiDAR point cloud can be corrupted, for example, due to sensor malfunction or environmental factors. It has been shown in [8, 9] that, even seemingly insignificant perturbations, like introduction of jitter or minute amount of noise to the point cloud, can significantly decrease the performance of several state-of-the-art 3D object recognition architectures. This lack of robustness can limit the utility of 3D recognition in numerous applications, including in construction industry, geo-surveying, manufacturing and autonomous driving. Distribution shifts that can affect 3D data are diverse in nature and it might not be feasible to train the network for all the shifts which can possibly be observed in point clouds at test-time. Thus, there is a need to adapt to these shifts online at test-time, in an unsupervised manner.

Test-Time Training (TTT) leverages unlabeled test data to adapt the classifier to the change in data distributions at test-time in an online manner. Several TTT approaches have been recently proposed for the 2D image domain. The main techniques include regularizing the classifier on test data with objective functions defined on the entropy of its predictions [4, 11, 13], updating the statistics of the batch normalization layers to match the distribution of the test data [6], and training the network on test data with self-supervised tasks [5, 10]. However, existing 2D TTT methods fail when naively applied to the 3D point clouds, stressing upon the need for 3D-specific TTT methodologies, which are currently non-existent.

In this paper, we address the problem of test-time training for 3D point cloud classification. We propose a 3D-specific method, MATE, which adopts the self-supervised paradigm [5, 10], in which a deep network is adapted by solving a self-supervised task for the OOD test data. Our choice is dictated by the availability of a self-supervised task that perfectly matches our goal of adapting 3D networks. Masked autoencoder proved very effective in pre-training 3D object recognition networks [7], and adapting deep networks to corruptions of 2D images [1]. It removes a large portion of the point cloud, and tasks the network with reconstructing the entire point cloud given only the part that has not been removed. We use this procedure to update the network on every test sample that is used for the adaptation.

Our main contributions are extending TTT to the 3D domain and showing that simply adopting TTT techniques widely used in the 2D image domain is not a viable solution for 3D, stressing out the need for 3D-specific approaches. To this end, we demonstrate how well-suited and powerful

[†] Equally contributing authors.
Correspondence: muhammad.mirza@icg.tugraz.at

masked autoencoding is to address online test-time training for 3D data. We conduct extensive evaluations on three point cloud recognition datasets. Apart from achieving strong performance gains for online adaptation, we discover and highlight several useful properties for TTT with masked autoencoders. For example, our MATE achieves significant performance gains even when masking 95% of tokens from point clouds. This seemingly nuance can have important benefits: At test-time, the encoder only needs to process the remaining 5% of the visible tokens to adapt the network, radically limiting the computational overhead of the adaptation.

2. MATE

We first describe our problem setting and model architecture in detail, then we describe our training setup and finally provide details about our test-time training methodology.

2.1. Problem setting

We follow the conventional test-time training setting, proposed by TTT [10], where at test-time we first adapt on a single sample and then test it. For adaptation we use the MAE reconstruction task. To process the point clouds, we use the PointMAE [7]. Given a point cloud $\mathcal{X}=\{\mathbf{p}_i\}_{i=1}^{N}$ of N points $\mathbf{p}_i=(x,y,z)^T$, the points are grouped into tokens, that is, possibly overlapping subsets of nearby points, using the farthest point sampling [7]. A proportion of tokens equal to the mask ratio m is then randomly masked, yielding the masked tokens, that we denote by $\mathcal{X}^m$, while $\mathcal{X}^v$ represent the remaining visible tokens. During joint training, we assume access to the training data $\mathcal{S}=\{(\mathcal{X},\mathcal{Y})\}$, where each point cloud $\mathcal{X}$ is accompanied by its ground truth label $\mathcal{Y}$. During test-time training, we do not have access to the entire test dataset but instead adapt to each single sample as it is encountered. After adapting the network parameters on each sample, the updated weights are used for predicting the class label. A detailed overview of different stages in our pipeline is shown in Figure 1, while the pseudocode is provided in the supplementary material.

2.2. Architecture

We adopt the PointMAE architecture [7], proven to work well in unsupervised pre-training for 3D object classification. It consists of an encoder E, a decoder D, a prediction head P, and a classifier head C. The encoder E consists of 12 standard transformer blocks and receives only the unmasked point patches as input. The decoder D is similar to E, however, it is lightweight (4 blocks), which makes the encoder-decoder structure asymmetrical. The masked point patches and the embeddings from the unmasked point patches are fed to the decoder after concatenation. The decoder feeds the embeddings to the prediction head P, which is a simple linear fully connected layer and reconstructs the points in coordinate space. The classifier head C is a projection from the dimensions of the encoder output to the number of classes in the respective dataset. We use 3 fully connected layers with ReLU non-linearity, batch normalization and dropout as our classification head.

2.3. Joint Training

Previous methods that employ the masked autoencoder for images or point clouds [1, 7] pre-train the encoder and decoder in a self-supervised manner and subsequently train the classifier on top of it. In contrast, to make the encoder learn embeddings that at the same time describe the input geometry and are well suited for the downstream task, we train the two heads jointly. Given all the parameters of the network $\{\theta_E,\, \theta_D,\, \theta_P,\, \theta_C\}$, the joint training is posed as

$$\min_{\theta_E,\,\theta_D,\,\theta_P,\,\theta_C} \mathbb{E}_{(\mathcal{X},\mathcal{Y})\in\mathcal{S}}\big[L_c(\mathcal{X},\mathcal{Y};\theta_E,\theta_C) + \lambda\cdot L_s(\mathcal{X};\theta_E,\theta_D,\theta_P)\big], \quad (1)$$

where the expectation is taken over the training set $\mathcal{S}$, and the hyper-parameter λ balances the two tasks. We set $\lambda=1$ for all experiments. Here, L_c is a cross entropy (CE) loss to learn the main classification task

$$L_c(\mathcal{X},\mathcal{Y};\theta_E,\theta_C) = CE(C\circ E(\mathcal{X}^v),\mathcal{Y}), \quad (2)$$

where $\mathcal{X}^v$ are the visible tokens and L_s is the self-supervised loss. Following [7], we use

$$L_s(\mathcal{X};\theta_E,\theta_D,\theta_P) = CD(P\circ D\circ E(\mathcal{X}^v),\mathcal{X}), \quad (3)$$

which is the Chamfer distance CD between the reconstructed tokens, and the training point sets $\mathcal{X}$.

2.4. Test-Time Training

Given the parameters $\{\theta_E,\, \theta_D,\, \theta_P,\, \theta_C\}$, trained jointly for the main classification task and the self-supervised reconstruction task on the training data. Our goal at test-time is to adapt to the OOD test data in an unsupervised manner, to achieve generalization. For this purpose we use the self-supervised MAE reconstruction task to adapt the network parameters to the OOD test sample.

For adaptation at test-time, we are granted access to only a single out-of-distribution point-cloud $\tilde{\mathcal{X}}$, without any ground truth label. The point cloud is tokenized and masked, and processed by the encoder E which yields the encoding vector. Finally, the patch encodings and the masked patches are concatenated and fed to the decoder D and ultimately to the prediction head P to obtain the reconstructed point cloud. The reconstruction loss is again an l_2 Chamfer distance between the reconstructed masked tokens and the corresponding ground truth tokens from the original out-of-distribution test sample. Our objective at test-time is to update the parameters of the encoder θ_E, decoder θ_D and the prediction head

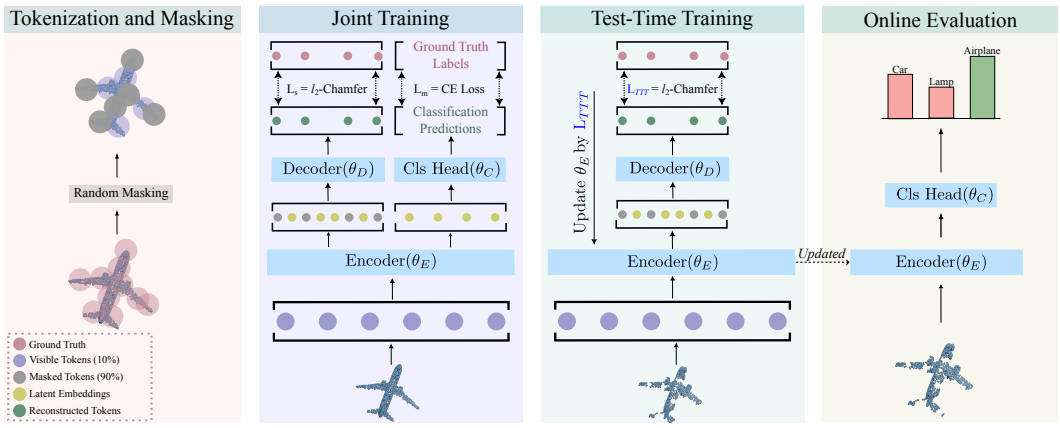

Figure 1: Overview of our 3D Test-Time Training methodology. We build on top of PointMAE. The input point cloud is first tokenized and then randomly masked. For our setup, we mask 90% of the point cloud. For joint training the visible tokens from the training data are fed to the encoder to get the latent embeddings from the visible tokens. These embeddings are fed to the classification head for the classification loss and concatenated with the masked tokens and fed to the decoder for reconstruction to obtain the reconstruction loss. Both losses are optimized jointly. For adaptation to an out-of-distribution test sample at test-time, we only use the MAE reconstruction task. Finally, after adapting the encoder on this single sample, evaluation is performed by using the updated encoder weights.

θ_P to generalize to the OOD test sample. More formally, for test-time training we minimize

$$L_{TTT} = \min_{\theta_E, \theta_D, \theta_P} L_s(\tilde{\mathcal{X}}; \theta_E, \theta_D, \theta_P). \tag{4}$$

Although for the downstream task of object classification, we only require the updated encoder, through experiments we find that updating the decoder and the prediction head does not affect the final classification performance.

2.5. Online Adaptation Variants

After adapting the encoder weights by the reconstruction loss during test-time training, prediction scores for the OOD sample are obtained by using the classifier head C, from the joint training phase. Following TTT [10], we provide two variants of our MATE, which are described as follows:

MATE-Standard only assumes access to a single point cloud sample at test-time and the goal is to iteratively adjust the weights on single samples in order to make the right prediction. For this purpose, we perform 20 gradient steps on the encoder parameters θ_E to minimize the objective in Eq. (4), computed for one test sample. As the next sample is received, we reinitialize the weights for all the parameters $\{\theta_E, \theta_D, \theta_P\}$, and repeat the same process again.

MATE-Online assumes that point clouds are received in a stream. For this version, we accumulate the model updates after adaptation on each sample. We only calculate (and backpropagate) L_{TTT} from Eq. (4), once for each sample.

2.6. Augmentations

During joint-training we only train the network with point cloud scale and translation augmentations, as originally used by the authors of PointMAE. For test-time training, we do not use any augmentation, instead we construct a batch (following [10]) from the single point cloud sample and for reconstruction, we randomly mask 90% of the tokens. Random masking is essential for MAE and also provides us with a natural augmentation. We further find that we can increase the masking ratio up to 95% and still get an impressive performance improvement. This is in contrast to images where a masking ratio of up to $70 - 75\%$ is employed. Higher masking ratios help in efficient test-time training, since only the unmasked tokens are processed by the encoder, which carries the majority of the computation effort because it has a larger structure than the decoder.

3. Experimental Evaluation

We provide results for both the Standard and Online evaluation variants. For implementation details, extended evaluations and ablation studies, please refer to the supplementary.

3.1. Datasets

We test MATE on the task of object classification for 3D point clouds. To this end, we use one of the popular object classification datasets, ModelNet40-C.

ModelNet-40C. ModelNet-40C [9] is a benchmark for evaluating robustness of point cloud classification architec-

corruptions:	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
Source-Only	66.6	59.2	7.2	31.7	74.6	67.7	69.7	59.3	75.1	74.4	38.1	53.7	70.0	38.6	23.4	53.9
Joint-Training	62.4	57.0	32.0	58.8	72.1	61.4	64.2	75.1	80.8	67.6	31.3	70.4	64.8	36.2	29.1	57.6
DUA	65.0	58.5	14.7	48.5	68.8	62.8	63.2	62.1	66.2	68.8	46.2	53.8	64.7	41.2	36.5	54.7
TTT-Rot	61.3	58.3	**34.5**	48.9	66.7	63.6	63.9	59.8	68.6	55.2	27.3	54.6	64.0	40.0	29.1	53.0
SHOT	29.6	28.2	9.8	25.4	32.7	30.3	30.1	30.9	31.2	32.1	22.8	27.3	29.4	20.8	18.6	26.6
T3A	64.1	62.3	33.4	65.0	75.4	63.2	66.7	57.4	63.0	72.7	32.8	54.4	67.7	39.1	18.3	55.7
TENT	29.2	28.7	10.1	25.1	33.1	30.3	29.1	30.4	31.5	31.8	22.7	27.0	28.6	20.7	19.0	26.5
MATE-Standard	75.0	71.1	27.5	67.5	78.7	69.5	72.0	**79.1**	84.5	75.4	44.4	73.6	72.9	39.7	34.2	64.3
MATE-Online	**82.9**	**80.6**	32.4	**74.0**	**85.7**	**78.3**	**80.2**	78.1	**86.5**	**79.3**	**56.6**	**77.9**	**77.1**	**49.7**	**50.0**	**71.3**

Table 1: Top-1 Classification Accuracy (%) for all distribution shifts in the ModelNet-40C dataset. All results are for the PointMAE backbone trained on clean train set and adapted to the OOD test set with a batch-size of 1. *Source-Only* denotes its performance on the corrupted test data without any adaptation. Highest Accuracy is in bold, while second best is underlined.

tures. In this benchmark, 15 common types of corruptions are induced on the original test set of ModelNet-40 [12]. These corruptions are divided into 3 parent categories comprising *transformation*, *noise* and *density*. Their goal is to mimic distribution shifts which occur in real-world, *e.g.*, common noise patterns on a LiDAR scan due to fault in the sensors capturing the data.

3.2. Baselines

We compare our MATE to several other TTT approaches proposed for images. In our work we assume access to only a single sample for adaptation at test-time, thus, for a fair comparison with our MATE, we also test other baselines in the single sample adaptation protocol. However, many 2D baselines fail in the single sample protocol, thus, we also provide results for larger batch sizes. A brief description of all the baselines is as follows.

- *Source Only* refers to the PointMAE backbone trained in a supervised manner on the classification task only. For testing on the OOD data, we do not mask the tokens, instead feed the entire point cloud.
- *Joint Training* [2] results are obtained by training the network jointly on the classification and MAE reconstruction task and testing it on the target data (*e.g.* ModelNet-40C) without adaptation.
- *SHOT* [4] proposes to minimize the expected entropy of predictions calculated from the output probability distribution from the network.
- *T3A* [3] relies on learning class specific prototypes to replace the classifier which is learned on the training set.
- *TENT* [11] also minimizes the entropy of predictions from the output of the classifier.
- *DUA* [6] updates the batch normalization statistics to adapt to OOD test images at test-time.
- *TTT-Rot* [10] with self-supervised rotation prediction task proposes to adapt to test data at test-time by predicting the rotation of images. Following the original paper, we train a network for classification and rotation prediction tasks.

Method	Source	TENT	SHOT	T3A	MATE-O
Accuracy (%) (BS - 128)	53.9	65.6	63.8	55.9	74.5

Table 2: Mean Top-1 Classification Accuracy (%) for ModelNet-40C by using a larger batch size (BS) of 128 for baselines and MATE-Online.

3.3. Results

ModelNet-40C: In Table 1 we provide the results for all the distribution shifts in the ModelNet-40C dataset. From the table, we see that our MATE outperforms other baselines comfortably. Furthermore, even our MATE-Standard performs better than the baselines with a considerable margin, while also performing favorably on individual distribution shifts. The test-time training approaches which rely on post-hoc regularization, *e.g.* SHOT [4] and TENT [11] perform poorly, while T3A [3] is only marginally above Source-Only baseline. This shows that the approaches designed for image data cannot be trivially transferred to the 3D domain. Moreover, all these approaches require larger batch sizes to work in the 2D domain. These approaches cannot adapt on a single test sample at test-time. For example, the entropy based approaches [4, 11], can have a trivial solution while optimizing the entropy of a single test sample. For larger batch sizes, we see that SHOT, TENT and T3A show some improvement in results (Table 2) but still MATE outperforms them comfortably. However, we reason that in online real-time applications we cannot access a batch of test data for adaptation, thus it is necessary that the TTT approaches work well even while having access to a single sample for adaptation at test-time.

4. Conclusion

Test-time training approaches designed for the 2D image domain can often degrade significantly if naively applied to the 3D data, requiring specialized 3D-specific designs. To this end, we show that masked autoencoding is a powerful self-supervised objective, which makes the network robust to various distribution shifts occurring in 3D point clouds.

References

[1] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-Time Training with Masked Autoencoders. In *NeurIPS*, 2022. 1, 2

[2] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using Pre-Training Can Improve Model Robustness and Uncertainty. In *Proc. ICML*, 2019. 4

[3] Yusuke Iwasawa and Yutaka Matsuo. Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization. In *NeurIPS*, 2021. 4

[4] Jiashi Feng Jian Liang, Dapeng Hu. Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation. In *Proc. ICML*, 2020. 1, 4

[5] 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? In *NeurIPS*, 2021. 1

[6] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The Norm Must Go On: Dynamic Unsupervised Domain Adaptation by Normalization. In *Proc. CVPR*, 2022. 1, 4

[7] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked Autoencoders for Point Cloud Self-supervised Learning. In *Proc. ECCV*, 2022. 1, 2

[8] Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and Analyzing Point Cloud Classification under Corruptions. In *Proc. ICML*, 2022. 1

[9] Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z Morley Mao. Benchmarking Robustness of 3D Point Cloud Recognition Against Common Corruptions. In *Proc. ICLR*, 2022. 1, 3

[10] 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 *Proc. ICML*, 2020. 1, 2, 3, 4

[11] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully Test-time Adaptation by Entropy Minimization. In *Proc. ICLR*, 2020. 1, 4

[12] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Proc. CVPR*, 2015. 4

[13] Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test Time Robustness via Adaptation and Augmentation. In *NeurIPS*, 2021. 1