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. Our experiments show that MATE also achieves competitive performance by adapting sparsely on the test data, which further reduces its computational overhead, making it ideal for real-time applications.

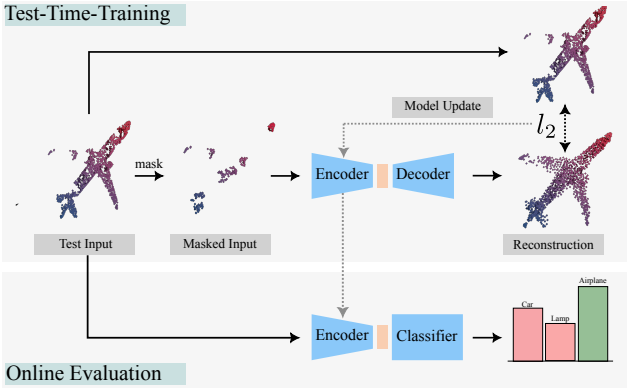

Figure 1: Overview of our Test-Time Training methodology. We adapt the encoder to a single out-of-distribution (OOD) test sample online by updating its weights using a self-supervised reconstruction task. We then use the updated weights to make a prediction on the test sample. To enable this approach, the encoder, decoder, and the classifier are co-trained in the classification and reconstruction tasks [17], which is not shown in the figure.

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 [19, 22] 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

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

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 [12, 26, 30], updating the statistics of the batch normalization layers to match the distribution of the test data [16], and training the network on test data with self-supervised tasks [14, 23]. 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 [14, 23], 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 [17], and adapting deep networks to corruptions of 2D images [6]. 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. An overview is provided in Figure 1.

Our main contributions are extending TTT to the 3D point cloud 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 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 the 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. The overhead from TTT can be further reduced by adapting sparsely to test data, as MATE can achieve significant performance gains over un-adapted networks by only adapting on every 100-th sample of the OOD test data.

2. Related works

Our work is related to Unsupervised Domain Adaptation (UDA), Self-Supervised Learning (SSL) and more closely to methods which learn on test instances.

Unsupervised Domain Adaptation. UDA methods aim to bridge the domain gap between the source and target domains without requiring access to labels from the target domain. UDA has gained considerable traction in the 3D vision community. PointDAN [18] aligns local and global point cloud features from the source and target domain in an end-to-end manner. Liang *et al*. [13] propose to predict masked local structures by estimating cardinality, position and normals for the point cloud. Shen *et al*. [20] first propose to encode the underlying geometry of point clouds from the target data with the help of implicit functions and resort to pseudo-labeling in the second step. For 3D object detection, adversarial augmentation is proposed by 3D-VField [20] for generalization to different domains. MLC-Net [15] proposes to use a student-teacher network along with pseudo-labeling. Wang *et al*. [27] propose to bridge the domain gap for 3D object detection by using priors, such as bounding box sizes from the target domain. Although unsupervised domain adaptation approaches tackle an important problem, they assume knowledge about the test distribution and try to mitigate the distribution mismatch by an extensive training phase. On the other hand, test-time training requires no such priors and offers a setting which is more closer to real world scenarios, where on-the-fly adaptation is required.

Self-Supervised Learning. Self-Supervised representation learning thrives on the idea of extracting supervision from the data itself. A popular SSL training objective is to bring the representations from the two randomly augmented views from the same sample closer and push apart the views from the other samples in the batch [3, 4, 10, 29]. Another approach for SSL is to extract the supervision from the reconstruction of the input data. Self-supervised representation learning by using Autoencoders [25] has been a long-standing research topic in computer vision. Recently, He *et al*. [8] proposed Masked Autoencoders (MAE) for self-supervised representation learning in the image domain. MAE uses an asymmetric encoder-decoder structure based on the Vision Transformer [5]. High proportion of the image tokens ($70-75\%$) are masked and the SSL objective is to reconstruct the masked tokens. On a similar note, Pang *et al*. [17] propose Point-MAE, an MAE framework for self-supervised representation learning in 3D point cloud domain and show that due to the sparse nature of point clouds, a more severe masking ratio can also be employed. In our work we also use reconstruction of point clouds as an auxiliary self-supervised task for test-time training. To this end, we use the PointMAE framework and at test-time get our supervisory signal by reconstructing highly masked regions from the OOD input point cloud.

Test-Time Training. TTT methods can be divided in to two distinct groups. The first group of methods add post-hoc

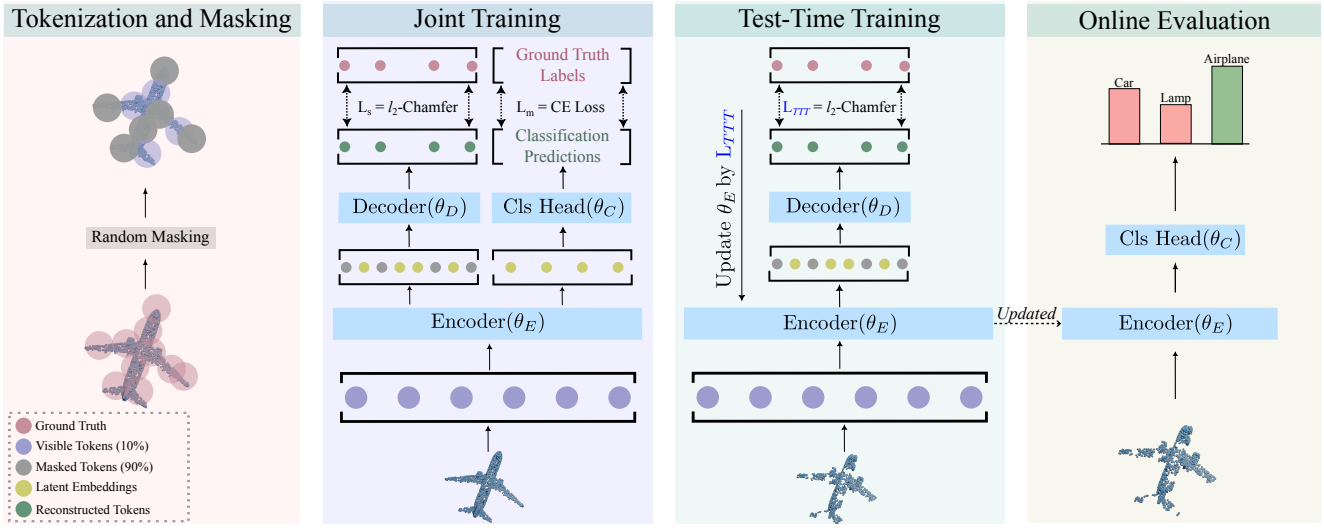

Figure 2: Overview of our 3D Test-Time Training methodology. We build on top of PointMAE. The input point cloud is first tokenized and then randomly masked. For our setup, we mask 90% of the point cloud. For joint training the visible tokens from the training data are fed to the encoder to get the latent 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.

regularization for adaptation to OOD test data. Boudiaf *et al*. [1] propose a gradient free TTT approach, which promotes consistency of output predictions coupled with Laplacian regularization. TENT [26], SHOT [12] and MEMO [30] rely on entropy minimization from the output softmax distribution. T3A [11] casts TTT as a prototype learning problem, while DUA [16] employs online statistical correction in the batch normalization layers for TTT. We test several of these approaches by porting them for TTT in the 3D point cloud recognition task but none of these approaches prove to be a competitive baseline for our MATE (Section. 4.4), further highlighting the need for 3D-specific methods.

The other group of methods propose to use auxiliary self-supervised tasks for adaptation to distribution shifts at test-time and are more closely linked to our MATE. Sun *et al*. [23] employ rotation prediction [7] as an auxiliary task for TTT. TTT++ [14] uses contrastive self-supervised learning (SimCLR [3]) as an auxiliary objective. TTT-MAE [6] substitutes the self-supervised objective with Masked Autoencoder [8] reconstruction task for TTT in the image domain. A general insight from these works implies that the choice of auxiliary self-supervised task is of utmost importance. MATE also employs the task of masked auto-encoding to drive the adaptation, but it reconstructs point clouds instead of images. This forces the network to encode the geometry of the point cloud and model long-range dependencies between local shapes. Furthermore, our experiments show that, for 3D point clouds, geometric reconstruction is a better auxiliary task than rotation prediction, which is employed by TTT [23].

3. 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.

3.1. Problem setting

We follow the conventional test-time training setting, proposed by TTT [23], 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 [17]. 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 [17]. 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 2, while the pseudocode is provided in the supplementary material.

3.2. Architecture

We adopt the PointMAE architecture [17], 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.

3.3. Joint Training

Previous methods that employ the masked autoencoder for images or point clouds [6, 17] 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 [17], 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 $\mathcal{X}$.

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

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

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.

3.5. 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 [23], 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.

3.6. 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 [23]) 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.

4. Experimental Evaluation

We provide results for both the Standard and the Online evaluation variants. Here, we first describe the datasets we use for evaluation, second we provide our implementation details and later present our results.

4.1. Datasets

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

ModelNet-40C. ModelNet-40C [22] is a benchmark for evaluating robustness of point cloud classification architectures. In this benchmark, 15 common types of corruptions are induced on the original test set of ModelNet-40 [28]. 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.

ShapeNet-C. ShapeNetCore-v2 [2] is a large-scale point cloud classification dataset consisting of 51127 shapes from 55 categories. We divide this dataset into three splits, train (35789, 70%), validation (5113, 10%) and test (10225, 20%). We provoke 15 different corruptions in the test set of ShapeNet, similar to ModelNet-40C, by using the open source implementation provided by [22]. We refer to this dataset as ShapeNet-C.

ScanObjectNN-C. ScanObjectNN [24] is a point cloud classification dataset which is collected in the real-world. It consists of 15 categories with 2309 samples in the train set and 581 samples in the test set. We again use the open source code provided by [22] to cause 15 different corruptions in the test set of ScanObjectNN for our evaluations, which we refer to as ScanObjectNN-C.

4.2. Implementation Details

We jointly train a network for supervised classification and self-supervised reconstruction tasks, as described in Section 3.3. For joint training we only use 10% of the visible tokens for the self-supervised reconstruction and the classification task. However, to obtain the final classification scores at test-time, we always feed 100% of the tokens to the PointMAE backbone. For ModelNet-40 and ShapeNetCore experiments, we train the networks from scratch for 300 epochs with a learning rate of 0.001 and Cosine scheduler. ScanObjectNN is a small-scale dataset, thus, we finetune the PointMAE network pre-trained on the large-scale ShapeNet-55 [2] dataset with a learning rate of 0.0005 and a Cosine scheduler for only 100 epochs, to avoid overfitting. All these models (including the vanilla PointMAE) use only the point cloud scaling and translation as augmentations[1]. For a fair comparison, the architectural details for all baselines and our method are kept constant.

During test-time training we update the encoder, decoder and the prediction head only. The classification head remains frozen. We use a learning rate of $5\mathrm{e}{-5}$ for TTT on ModelNet-40C, a learning rate of $1\mathrm{e}{-4}$ for ShapeNet-C and ScanObjectNN-C. We use AdamW optimizer for both, pre-training and the test-time training. To calculate the test-time training loss, we construct a batch of 48 from the single corrupted point cloud at test-time and randomly mask 90% of each sample in the batch. To encourage reproducibility, our entire codebase and pre-trained models are available at this repository: https://github.com/jmiemirza/MATE.

4.3. 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* [9] 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* [12] proposes to minimize the expected entropy of predictions calculated from the output probability distribution from the network.
- *T3A* [11] relies on learning class specific prototypes to replace the classifier which is learned on the training set.
- *TENT* [26] also minimizes the entropy of predictions

[1]We avoid other augmentations, *e.g.* jitter or rotation, because they might correlate with the corruptions in the ModelNet-C benchmark and can provide us with an unfair advantage during TTT.

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 (copied 48 times through random masking). *Source-Only* denotes its performance on the corrupted test data without any adaptation. Highest Accuracy is in bold, while second best is underlined.

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.

from the output of the classifier.

- *DUA* [16] updates the batch normalization statistics to adapt to OOD test images at test-time.

- *TTT-Rot* [23] 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.

4.4. 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 [12] and TENT [26] perform poorly, while T3A [11] 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 [12, 26], 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.

From the results we also see that the mean performance over all corruptions of TTT-Rot falls below Source-Only, even though it is originally designed for the single sample adaptation scenario in the 2D domain. This could be an indication that the rotation prediction task is not well suited for test-time adaptation for 3D data. However, for Background corruption TTT-Rot [23] fares well. This might be because Background corruption introduces artifacts in the background and TTT-Rot uses the entire point cloud for test-time adaptation, so it can adapt to this corruption better. On the other hand, we only adapt with 10% of the visible tokens and might not be able to capture these artifacts introduced in the background. Furthermore, we analyze the reconstructions from the background corruption and find that the reconstruction results are worse as compared to other corruptions. We show these visualizations in the supplemental. These reconstruction results suggest that the reconstruction task is co-related with the classification task. Hence, better reconstruction accounts for better adaptation performance. We also see a similar trend for the TTT loss and classification accuracy at each adaptation step for corruptions in the ModelNet-40C. These results are also delegated to the supplementary material.

ShapeNet-C: In Table 3 we provide Top-1 Accuracy (%) for object classification on the ShapeNet-C dataset. We again see that both evaluation variants of our MATE show impressive results on the large-scale ShapeNet dataset. MATE-Online has a huge performance gain over other baselines, which is expected, since for these evaluations we accumulate the model updates. Similarly, MATE-Standard also

corruptions:	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
Source-Only	69.2	62.8	10.3	56.2	70.1	70.5	71.9	85.5	86.2	73.9	41.3	84.4	69.9	7.9	3.9	57.6
Joint-Training	72.5	66.4	15.0	60.6	72.8	72.6	73.4	85.2	85.8	74.1	42.8	84.3	71.7	8.4	4.3	59.3
DUA	76.1	70.1	14.3	60.9	76.2	71.6	72.9	80.0	83.8	77.1	57.5	75.0	72.1	11.9	12.1	60.8
TTT-Rot	74.6	72.4	23.1	59.9	74.9	73.8	75.0	81.4	82.0	69.2	49.1	79.9	72.7	14.0	12.0	60.9
SHOT	44.8	42.5	12.1	37.6	45.0	43.7	44.2	48.4	49.4	45.0	32.6	46.3	39.1	6.2	5.9	36.2
T3A	70.0	60.5	6.5	40.7	67.8	67.2	68.5	79.5	79.9	72.7	42.9	79.1	66.8	7.7	5.6	54.4
TENT	44.5	42.9	12.4	38.0	44.6	43.3	44.3	48.7	49.4	45.7	34.8	48.6	43.0	10.0	10.9	37.4
MATE-Standard	77.8	74.7	4.3	66.2	78.6	76.3	75.3	**86.1**	**86.6**	79.2	56.1	84.1	76.1	12.3	13.1	63.1
MATE-Online	**81.5**	**78.6**	**40.9**	**75.9**	**81.6**	**79.7**	**80.1**	84.9	85.9	**81.8**	**70.8**	**85.1**	**79.0**	**14.2**	**16.6**	**69.1**

Table 3: Top-1 Classification Accuracy (%) for all distribution shifts in the ShapeNet-C dataset. All results are for the PointMAE backbone trained on clean train set set and adapted to the OOD test set with a batch-size of 1.

Method	Accuracy (%)	Method	Accuracy (%)
Source	45.7	TTT-Rot	46.1
SHOT	38.3	T3A	40.3
JT	45.6	MATE-S	47.0
DUA	46.0	MATE-O	**48.5**

Table 4: Top-1 Classification Accuracy (%) averaged over the 15 corruptions in the ScanObjectNN-C dataset (adapted with batch size 1). JT: Joint Training, MATE-S: MATE-Standard, MATE-O: MATE-Online

	Mask Ratio (%)					
	97.5	95	90	80	70	60
MATE Online	56.9	71.6	71.3	71.5	71.6	71.5

Table 5: Top-1 Classification Accuracy (%) averaged over all corruptions in the ModelNet-40C dataset, while using different masking ratios for test-time training. The accuracy for Source-Only baseline is 57.6%.

outperforms other baselines and even surpasses MATE-Online on the density-related corruptions of the point clouds. We again notice that popular 2D test-time training methods [11, 12, 16, 23, 26] struggle for the ShapeNet dataset as well. These results further strengthen our reasoning that the need for 3D test-time training cannot be fulfilled by naively porting the 2D TTT approaches.

ScanObjectNN-C: We also test our MATE on point clouds collected in real world, on which we introduce the corruptions proposed in the ModelNet-C benchmark [22]. The results are provided in Table 4 and are in-line with the other datasets. These results show the applicability of MATE on data collected in the real world scenarios as well.

	Batch Size for Test-Time Training							
	1	2	8	16	24	32	40	48
MATE Online	43.1	66.4	69.7	70.2	70.4	70.5	70.5	71.3

Table 6: The effect of batch size for TTT. We provide the Mean Top-1 Accuracy (%) over all the corruptions in the ModelNet-40C dataset for different batch sizes used for TTT. The accuracy for Source-Only baseline is 57.6%.

5. Ablation Studies

We additionally test how MATE performs with different masking ratios, scenarios where sparse adaptation on test samples is required, the effect of batch size on TTT and the effect on performance while combining multiple corruption types together.

5.1. Masking Ratios

The PointMAE has an asymmetric encoder-decoder design. The decoder is a lightweight architecture, while the encoder is a deeper network. Therefore, most of the computation effort is spent in the encoding part of the pipeline. Since the encoder processes only the visible tokens, higher masking ratio implies lower burden for the encoder. We find that our MATE can work with extremely high masking ratios, making test-time training very efficient. The results for adaptation with different masking ratios are provided in Table 5. We see that even with a severe masking of 95% of the tokens (*i.e.* only processing 5% visible tokens), our MATE can achieve 14 percent-points over the Source-Only (without adaptation) results. Even with 97.5% masking, we still improve on the Source-Only results. These results also show that lower masking ratios do not give us more gain in performance but instead could induce latency during test-time training, undesirable for real-time applications.

	Source	JT	DUA	TTT-Rot	MATE-S	MATE-O
Comb - 1	33.9	36.7	42.6	34.3	47.7	**55.7**
Comb - 2	29.6	34.7	40.6	32.9	45.2	**51.4**
Comb - 3	28.3	33.3	41.5	30.7	44.5	**52.5**
Mean	30.6	34.8	41.6	32.6	45.8	**53.2**

Table 7: Top-1 Mean Accuracy (%) for three different datasets constructed by combining 2 randomly chosen corruptions for each subsequent sample in the test-set of ModelNet-40. JT: Joint Training, MATE-S: MATE-Standard, MATE-O: MATE-Online

5.2. Strides for TTT

Some applications might require adaptation at test-time with minimum latency. For example, a test-time training method deployed in autonomous vehicles would ideally be required to adapt at a high frame-rate per second (FPS). Thus, a test-time training method should be able to run with *close to real-time* adaptation speed. Since most of the computation overhead for adaptation methods is during the backward pass, adapting to test samples sparsely should help to reduce the computation effort. In order to scratch the boundaries of our MATE for achieving a higher FPS, we design an experiment where we only adapt at test-time after a certain number of samples (stride). Results for ShapeNet dataset in this scenario are provided in Figure 3. When performing an adaptation step after a stride, we find that our MATE can achieve close to real-time performance, with a minimum performance penalty. For example, when we take a gradient step on every 5-th sample, MATE can adapt at 20 FPS on an NVIDIA 3090 (for reference 30 FPS is often considered as real-time [21]) with only $\sim$3 percent-point drop in performance while comparing with the results obtained with a stride of 1 (adapting on each incoming sample). We can even increase the stride up to 300 and still achieve $\sim$3 percent-point better performance than the Source-Only results, with an FPS of 62. These results indicate the efficient nature of our MATE and its ability to show effective real-time adaptation performance. Results for ModelNet-40C in this adaptation protocol are provided in the supplementary.

5.3. Batch Size for Test-Time Training

MATE constructs a batch of 48 from each point cloud encountered at test-time for adaptation. The point cloud in this batch is randomly masked and then masked patches are reconstructed. Random masking helps us achieve a natural augmentation during test-time training. To test the effect of our design choice on the test-time training performance, we experiment with different batch sizes on the ModelNet-40C dataset. These results are provided in Table 6. Surprisingly, for batch size of 1, test-time adaptation performance

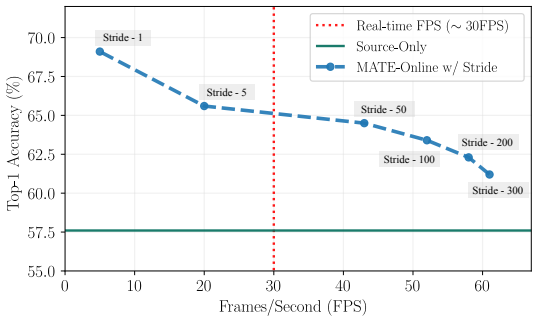

Figure 3: MATE can achieve real-time adaptation performance with only a minor performance penalty. Here, we report the Mean Top-1 Accuracy (%) over the 15 corruptions in the ShapeNet-C dataset for different adaptation strides. Strides represent the number of samples after which an adaptation step is performed.

falls below Source-Only but is 8.8 percent-point better than Source-Only for the batch size of 2. We also see that batch size larger than 8 achieve minor gains, thus it could be a resource-efficient alternative.

5.4. Combination of Distribution Shifts

In realistic scenarios there could be situations where the test sample might be corrupted with a combination of corruptions. Thus, a test-time training method should be able to cope with such scenarios as well. To test our MATE in such a scenario, we design an experiment where we randomly combine 2 corruption types (from the ModelNet-40C benchmark) for each sample in the test set of ModelNet-40 and create 3 such datasets. To generate these datasets, we ensure that all 15 corruption types are selected for each dataset and for each sample 2 corruptions are chosen randomly from the set of 15 corruptions. We test our MATE and other baselines on these datasets and provide the results in Table 7. We see that MATE can effectively adapt to this scenario as well and outperforms other baselines by a considerable margin. DUA fares better than TTT-Rot, because DUA does not use any geometric information, which is another indication that rotation prediction might not be a suitable test-time training objective for 3D point clouds.

5.5. Limitation

In this paper we propose the first TTT method for 3D point cloud data. To this end, we tested our MATE rigorously for the point cloud classification task. Focusing on this task we were able to show that masked autoencoders can provide extremely powerful self-supervisory signal for this task. However, application of TTT to other downstream tasks is out-of-scope for this work and thus we leave it for future exploration.

6. 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 are the first to propose a 3D test-time training method, MATE. We show that masked autoencoding is a powerful self-supervised auxiliary objective, which can make the network robust to various kinds of distribution shifts occurring in 3D point clouds. Our MATE, is computationally cheap and can also run in real-time adaptation scenarios while achieving significant performance gains.

Acknowledgment

We gratefully acknowledge the financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, Christian Doppler Research Association and Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-02068, Artificial Intelligence Innovation Hub). This work was also partially funded by the FWF Austrian Science Fund Lise Meitner grant (M3374) and Austrian Research Promotion Agency (FFG) under the projects High-Scene (884306) and SAFER (894164).

References

[1] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free Online Test-time Adaptation. In *Proc. CVPR*, 2022.

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proc. ICML*, 2020.

[4] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *Proc. CVPR*, 2021.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. ICLR*, 2020.

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

[7] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *Proc. ICLR*, 2018.

[8] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *Proc. CVPR*, 2022.

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

[10] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal Self-Supervised Representation Learning for 3D Point Clouds. In *Proc. CVPR*, 2021.

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

[12] 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.

[13] Hanxue Liang, Hehe Fan, Zhiwen Fan, Yi Wang, Tianlong Chen, Yu Cheng, and Zhangyang Wang. Point Cloud Domain Adaptation via Masked Local 3D Structure Prediction. In *Proc. ECCV*, 2022.

[14] 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.

[15] Zhipeng Luo, Zhongang Cai, Changqing Zhou, Gongjie Zhang, Haiyu Zhao, Shuai Yi, Shijian Lu, Hongsheng Li, Shanghang Zhang, and Ziwei Liu. Unsupervised Domain Adaptive 3D Detection with Multi-Level Consistency. In *Proc. CVPR*, 2021.

[16] 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.

[17] 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.

[18] Can Qin, Haoxuan You, Lichen Wang, C-C Jay Kuo, and Yun Fu. PointDAN: A Multi-Scale 3D Domain Adaption Network for Point Cloud Representation. In *NeurIPS*, 2019.

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

[20] Yuefan Shen, Yanchao Yang, Mi Yan, He Wang, Youyi Zheng, and Leonidas J Guibas. Domain Adaptation on Point Clouds via Geometry-Aware Implicits. In *Proc. CVPR*, 2022.

[21] Guangsheng Shi, Ruifeng Li, and Chao Ma. Pillarnet: Real-time and high-performance pillar-based 3d object detection. In *Proc. ECCV*, 2022.

[22] 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.

[23] 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.

[24] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In *Proc. ICCV*, 2019.

[25] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proc. ICML*, 2008.

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

[27] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in Germany, Test in The USA: Making 3D Object Detectors Generalize. In *Proc. CVPR*, 2020.

[28] 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.

[29] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *Proc. ICML*, 2021.

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