—Supplementary Material—
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.

In the following, we present the detailed Algorithm for MATE (Section 1), provide specifics for all the distribution shifts (Section 2), present experiments on ModelNet-40C achieving real-time test-time training (TTT) (Section 3) and show correlation between TTT and the auxiliary task of MAE reconstruction (Section 4).

1. Algorithm

In Algorithm. 1, we provide the detailed algorithm for our MATE, which consists of three phases: Joint training, Test-time training, and Online evaluation.

2. Details about Distribution Shifts

We use the corruption benchmark [4] to introduce 15 different types of commonly occurring distribution shifts on the test sets of the point cloud datasets we use for evaluation in our main manuscript. A description of these distribution shifts is provided as follows:

- *Uniform noise*: Random noise is added to each point in a point cloud, where the amount of noise is based on a uniform distribution and lie within a range of ± 0.05.

- *Gaussian noise*: Points are randomly perturbed and the amount of noise is based on a Gaussian (normal) distribution with values in range of ± 0.03.

- *Background noise*: Randomly add $(\frac{Number\ of\ Points}{20})$ points with values in the range of ± 1 in the bounding box of the point cloud.

- *Impulse noise*: Add a value in the range of ± 0.1 to a subset of the total number of points in the pointcloud.

- *Upsampling*: Additional points are added by duplicating the existing points in a point cloud.

- *RBF*: The point clouds are deformed based on the Radial Basis Function [1].

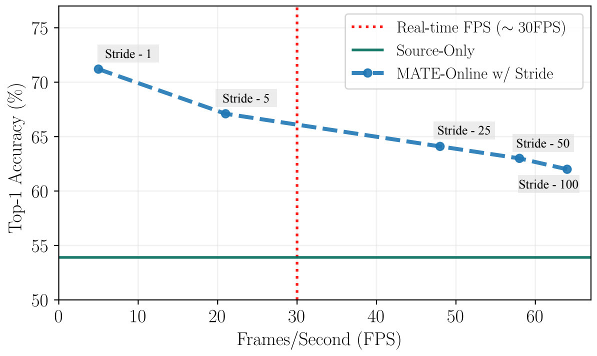

Figure 1: MATE can achieve real-time adaptation performance by only sacrificing some percent-points. Here, we report the Mean Top-1 Accuracy (%) over the 15 corruptions in the ModelNet-40C dataset for different adaptation strides. Strides represent the number of samples after which an adaptation step is performed.

- *Inverse_RBF*: To generate this shift, the Radial Basis Function and the resulting splines are inverted.

- *Local_Density_Decrease*: To generate this distribution shift, 5 local cluster centers and their 100 closest neighbors are chosen. Further, their point density is decreased by deleting $\frac{3}{4}$ of the points inside the clusters.

- *Local_Density_Increase*: Choose 5 local cluster centers with 100 closest neighbors. Then, keep these clusters, but randomly sample the rest of the pointcloud again with the original number of points. This results in double the density in the clusters, in comparison to the rest of the point cloud.

- *Shear*: Randomly compress and stretch the point cloud on the xy-plane. Here, the points get multiplied by values in

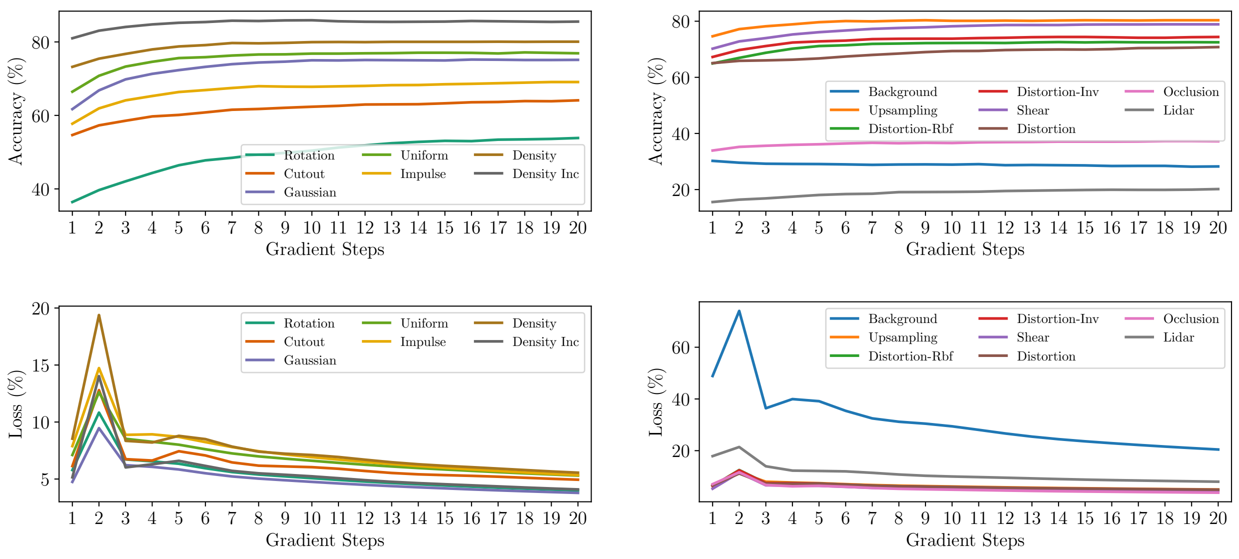

Figure 2: Accuracy (Top) and Reconstruction Loss (Bottom) for all corruption in the ModelNet-40C at each adaptation step for MATE-Standard. To avoid clutter, we split the different corruptions into two plots (left and right).

range of ±0.25 for each dimension.

- *Rotation*: Rotate all three spatial dimensions of a point cloud by a random angle in range of $\pm15°$.

- *Cutout*: To simulate cutout, generate 5 local clusters with 100 closest neighbors and remove these clusters from the original point cloud.

- *FFD*: For this distribution shift, the Free-Form Deformation (FFD) [3] is used. The point cloud is enclosed in a box consisting of splines which are defined by control points. The control points are shifted to deform the point cloud. A total of 125 control points are used with a deformation distance in a range of ±0.5.

- *Occlusion*: Occluded points are deleted by using ray-tracing from a random camera position. For this operation, precomputed meshes [6] are used.

- *LiDAR*: Point clouds are simulated as if they are generated from a LiDAR sensor. In addition to occlusion, inaccuracies based on reflections and noise are added.

3. Real-time Test-time Training

In the main manuscript (Figure 3), we provide results for MATE-Online while adapting sparingly to the test samples in ShapeNet-C dataset. We see that, while adapting sparingly on the test data, *i.e*. only back-propagating gradients after a certain number of samples (stride), our MATE can still achieve strong performance gains and can even match the real-time FPS (30), with only a minimum penalty on accuracy. Here, in Figure 1, we provide the results with different strides for the ModelNet-40C dataset, which is $\sim 4\times$ smaller than the ShapeNet-C dataset. While adapting sparingly, we see that, similar to the results on the large-scale ShapeNet-C, our MATE can also achieve close to *real-time* performance by dropping only a few percent-points as compared to adapting on each sample. For example, with a stride of 5 (adapting on every 5-th sample), our MATE drops only ~ 3 percent-points as compared to the results with stride-1 (adapting on each incoming sample), while obtaining an FPS of 21.

4. Classification and Reconstruction

At test-time, MATE adapts to each out-of-distribution (OOD) test sample by using the self-supervised reconstruction task as an auxiliary objective, leveraging masked autoencoders [2]. As each OOD sample is encountered, the network is adapted by the auxiliary self-supervised loss. This loss is an l_2 Chamfer distance between the reconstructed masked tokens and the corresponding ground truth tokens from the original OOD test sample. After adapting the network by back-propagating the gradients obtained from the auxiliary loss, the OOD sample is evaluated. In the main manuscript, we see that our test-time training methodology achieves strong performance gains on a variety of datasets for object classification in 3D point clouds. Naturally, the question arises – 'How a self-supervised task, *i.e*. reconstruction task, can help to adapt the network for a seemingly unrelated task, like object classification?' Through our experiments, we find that there is a correlation between the reconstruction task and the classification task and that is the reason for the improvement in classification accuracy by simply reconstructing the corrupted (OOD) test sample at test-time. We find this cor-

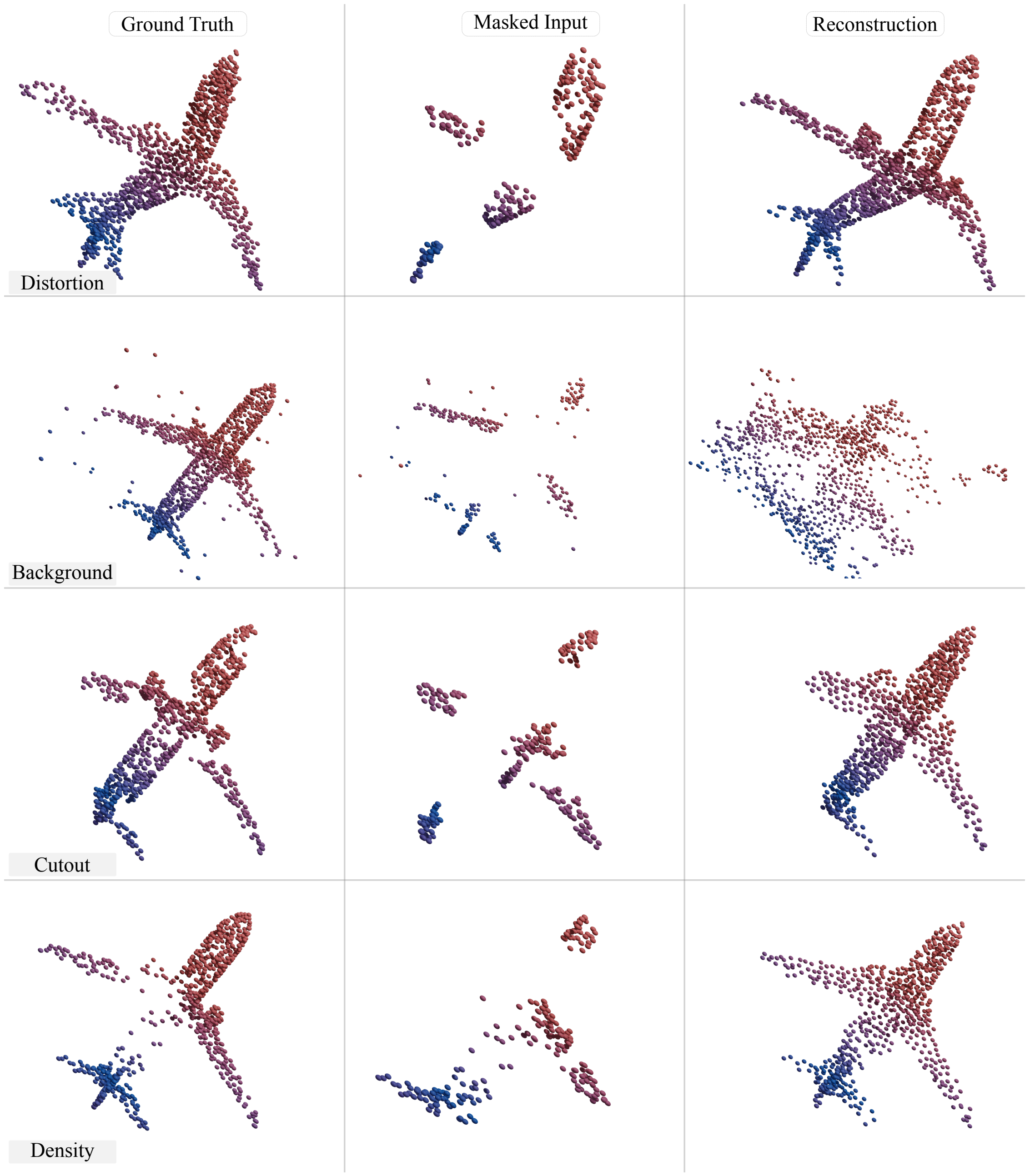

Figure 3: Reconstruction results for MATE-Standard at the 20-th gradient step for adaptation at test-time. We plot the out-of-distribution test sample for adaptation (left), 10% input visible tokens (center) and the corresponding reconstruction output (right) for four corruptions in the ModelNet-40C dataset.

relation empirically through two procedures, detailed in the following.

4.1. Loss and Accuracy

In the main manuscript, we test our MATE in two test-time training variants, described in Section 3.5, here we again provide a brief description in the interest of keeping the reading flow:

MATE-Standard assumes access to a single sample at test-time for adaptation. In order to adapt the network on this single sample, we take multiple gradient steps (*i.e.* 20) for test-time training. After adaptation on each sample, the network weights are re-initialized for adaptation on the next

Algorithm 1: Algorithm for MATE

```
Input: (Training data $\mathcal{S} = \{(\mathcal{X}, \mathcal{Y})\}$, Single
        out-of-distribution point-cloud $\tilde{\mathcal{X}}$)
1  begin
2      Define the network with encoder $E$, decoder $D$,
        prediction head $P$, classifier head $C$
3      Define the masking ratio $m$, batch size $b$, stride $s$
        and gradient steps $k$
4      # Joint Training.
5      for multiple epochs do
6          $\mathcal{X}^v$ = point-masking($\mathcal{X}, m$)
7          $L = CE(C \circ E(\mathcal{X}^v), \mathcal{Y})$ +
            $CD(P \circ D \circ E(\mathcal{X}^v), \mathcal{X})$ (Eq. (2, 3))
8          $L$.backward()
9          optimizer.step()
10     # Test-Time Training & Online Evaluation.
11     for idx, $\tilde{\mathcal{X}}$ in loader do
12         if idx % s == 0 then
13             $L_{TTT}$ = 0
14             for $k$ iterations do
15                 $\tilde{\mathcal{X}}^v$ = [point-masking($\tilde{\mathcal{X}}, m$) for _ in
                    range($b$)]
16                 $L_{TTT}$ += $CD(P \circ D \circ E(\tilde{\mathcal{X}}^v), \tilde{\mathcal{X}})$
                    (Eq. (4))
17             $L = L_{TTT}$.mean()
18             $L$.backward()
19             optimizer.step()
20         Evaluate $C \circ E(\tilde{\mathcal{X}}^v)$
```

sample.

MATE-Online assumes access to a stream of data for adaptation and the network updates are accumulated after adaptation on each sample in the stream. For this adaptation variant, we only take a single gradient step on each OOD test sample.

In Figure 2, we plot the Top-1 Accuracy on ModelNet-40C and the corresponding reconstruction loss at each gradient step for MATE-Standard. Please note that these results are plotted by taking the average of the accuracy over all the samples in the test set of ModelNet-40C at each gradient step. From the results it is evident that as the reconstruction loss decreases after each gradient step, the corresponding accuracy increases. This shows that as the model becomes better at reconstructing the OOD test sample, the classification performance is influenced in a positive way. We also see that there is a spike in the reconstruction loss during the initial update step. We hypothesize that this is because of the sudden distribution shift which is encountered at test-time, since the model is initially trained on clean point clouds. However, with more adaptation steps for test-time training, it slowly gets better at reconstructing the OOD sample.

Furthermore, an interesting result is that of the *Background* corruption. In the main manuscript, while listing the results for ModelNet-C (Section 4.4) we found that for the background corruption TTT-Rot [5] fares better than our MATE. From Figure 2, we see that for Background corruption the reconstruction loss is highest among all the other corruptions, that can be one of the reasons why MATE cannot perform well on this corruption. This also gives us an indication of the correlation between the reconstruction and the classification loss. To further investigate the background corruption and find more answers behind correlation of the two tasks, we visualize the reconstruction results next.

4.2. Reconstruction Results

We further analyzed the reconstruction results for different corruption types to get a deeper insight in to the correlation of the MAE reconstruction and the classification task. We find that for TTT with MATE-Standard, after 20 gradient steps, the reconstruction for the Background corruption is the worst as compared to reconstruction of other corruption types. We visualize these results for a few corruptions in the ModelNet-40C dataset for the $Airplane$ class in Figure 3. Reconstructions from the remaining corruptions also follow a similar pattern. Since MATE does not perform optimally for the Background corruption, this gives us an indication of the correlation between the auxiliary self-supervised reconstruction task and the downstream classification task. To conclude, our results show that if the auxiliary self-supervised reconstruction task is able to reconstruct the input corruption type optimally, MATE shows strong performance gains, which is an indication that these two tasks are correlated with each other.

References

[1] Davide Forti and Gianluigi Rozza. Efficient geometrical parametrisation techniques of interfaces for reduced-order modelling: application to fluid–structure interaction coupling problems. *International Journal of Computational Fluid Dynamics*, 2014.

[2] 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.

[3] Thomas W. Sederberg and Scott R. Parry. Free-Form Deformation of Solid Geometric Models. In *Proc. SIGGRAPH*, 1986.

[4] 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.

[5] 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.

[6] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing. *arXiv preprint arXiv:1801.09847*, 2018.