# Supplementary Material: Backpropagation-free Network for 3D Test-time Adaptation

Yanshuo Wang[1,2], Ali Cheraghian[2], Zeeshan Hayder[2], Jie Hong[1,2]*, Sameera Ramasinghe[5],
Shafin Rahman[3], David Ahmedt-Aristizabal[2], Xuesong Li[1,2], Lars Petersson[2], Mehrtash Harandi[2,4]
[1]Australian National University, [2]Data61-CSIRO, Australia
[3]North South University, Bangladesh, [4]Monash University, Australia
[5]Amazon, Australia
{yanshuo.wang, jie.hong, xuesong.li}@anu.edu.au,
{ali.cheraghian, zeeshan.hayder, david.ahmedtaristizabal, lars.petersson}@data61.csiro.au,
shafin.rahman@northsouth.edu, mehrtash.harandi@monash.edu, ramasisa@amazon.com

## Abstract

*Real-world systems often encounter new data over time, which leads to experiencing target domain shifts. Existing Test-Time Adaptation (TTA) methods tend to apply computationally heavy and memory-intensive backpropagation-based approaches to handle this. Here, we propose a novel method that uses a backpropagation-free approach for TTA for the specific case of 3D data. Our model uses a two-stream architecture to maintain knowledge about the source domain as well as complementary target-domain-specific information. The backpropagation-free property of our model helps address the well-known forgetting problem and mitigates the error accumulation issue. The proposed method also eliminates the need for the usually noisy process of pseudo-labeling and reliance on costly self-supervised training. Moreover, our method leverages subspace learning, effectively reducing the distribution variance between the two domains. Furthermore, the source-domain-specific and the target-domain-specific streams are aligned using a novel entropy-based adaptive fusion strategy. Extensive experiments on popular benchmarks demonstrate the effectiveness of our method. The code will be available at https://github.com/abie-e/BFTT3D.*

In this supplementary material, we provide additional experimental results and discussions.

*Corresponding author

## 1. Experiments

### 1.1. Main Results

**ShapeNet-C.** In this section, we further present the result of our method on a large dataset, ShapeNet-C [4]. ShapeNet [1] is a large-scale point cloud classification dataset collected from the real world. We use a subset of the complete ShapeNet dataset with clean 3D models, and it contains 55 distinct classes, with 35789 samples in the training set and 10225 in the test set. To build ShapeNet-C, we employ the setting proposed by [4] to generate different corruptions in the test set of ShapeNet with severity level 5. From Table 1, TENT [6] and BN [2] effectively reduce the classification error in test-time adaptation, and our approach, in particular, has the lowest error compared with other baseline approaches. This indicates that our approach is applicable to large-scale datasets, enhancing the source model's adaptability.

| Method | Mean ↓ |
|---|---|
| **Source** | 23.44 |
| **TENT** [6] | 21.45 |
| **BN** [2] | 22.05 |
| **SHOT [3]** | 26.94 |
| **BFTT3D (Ours)** | **21.34** |

Table 1. Experimental results on ShapeNet-C [4]. The mean classification errors in % are provided.

### 1.2. Ablation Study

**Number of prototypes.** The exact number of prototypes is determined based on the number of dataset samples. For

| Method | Inference time | Parameters | Mean ↓ |
|---|---|---|---|
| **MATE** [4] | 313 | 29.3M | 28.70 |
| **TENT** [6] | 24 | 2.3K | 73.50 |
| **BFTT3D (Ours)** | 19 | 0 | **19.23** |

Table 2. Analysis: model efficiency. The experiments are running on ModelNet-40C [5]. The inference time on each domain in $s$, the number of parameters that need backpropagation during the test time, and the mean error in % are given. The errors of MATE and TENT are provided in [4].

| Method | Time | Parameters | Mean ↓ |
|---|---|---|---|
| **MATE** [4] | 150.9 | 29.3M | 64.8 |
| **TENT** [6] | 10.5 | 2.3K | 58.18 |
| **BFTT3D (Ours)** | 9.7 | 0 | **54.46** |

Table 3. Analysis: model efficiency. The experiments are running on ScanObjectNN-C [4]. The inference time on each domain in $s$, the number of parameters that need backpropagation during the test time, and the mean error in % are given.

example, the prototype number is 2446 for ModelNet-40C, and the corresponding size of the stored features is 11.31 MB.

**Model efficiency.** In this part, we evaluate the model efficiency, including the inference time and number of parameters that need backpropagation, as summarized in Table 2. Since our BFTT3D does not require any test-time backpropagation, it has the least frequency of parameter updates, requiring 0 updates compared to other approaches. Thus, it consumes less time compared with other methods, like MATE [4] and TENT [6], in the adaptation stage. Other methods frequently perform parameter updates to perform adaptation, which adds an unavoidable computation burden in test time.

We present efficiency test results on ScanObjectNN-C [4] in Table 3. As shown in the table, MATE has the highest time consumption because it must perform sample-by-sample adaptation by constructing masked batches of samples in test-time. On the other hand, TENT requires less time in adaptation as it only needs to train the batch norm parameters in a batch-wise manner. In general, our BFTT3D has the least computational cost across the board because there are no model parameters that need to be back-propagated during adaptation, in contrast to other methods that require test-time training.

### 1.3. Discussion

**Statistical significance.** ModelNet-40C benchmark presents a minor domain gap, resulting in somewhat saturated performances (see Table 1 of the main paper). To showcase the statistical significance of our improvements,

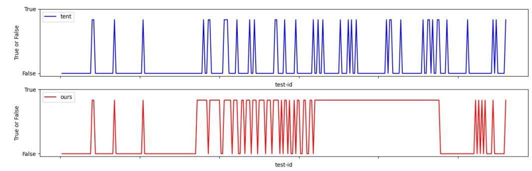


Figure 1. Forgetting and error accumulation. The blue line represents the prediction by sample ID, and the red line refers to our method.

we compute standard deviation (std) values on two cases. On ModelNet-40C and ScanObjectNN-C when employing PointNet, std values are $0.14\%$ and $0.21\%$ over five runs, respectively, indicating the significance of the results.

**Forgetting and error accumulation.** Forgetting and error accumulation issues originate from parameters requiring gradients. We give a comparison of ModelNet-40C under lidar corruption with baseline TENT, as shown in Figure 1. The blue line represents the prediction of TENT, and the red line refers to our method. Along with the adaptation, BFTT3D achieves better performance with fewer wrong predictions generated in the long run.

**Natural distribution shifts.** Based on results from both Tables 1 and 2 of the main paper, our network works well on domains like “background”, “occlusion”, and “lidar”, which are closer to natural distribution. In other words, our BFTT3D shows more robustness to the domain under natural distribution shifts.

**Clean accuracy.** The clean errors of Source-only/BFTT3D on ModelNet-40C are $9.76/9.61\%$, $8.91/8.64\%$, and $8.10/7.89\%$ for PointNet, DGCNN, and Curvenet. The clean results for ScanObjectNN-C are $22.55\%/20.93\%$, $15.32\%/14.25\%$, and $28.74\%/26.86\%$. Hence, we can see the benefit of our BFTT3D when there are no distribution shifts.

**Using the non-parametric network alone.** The logits from the non-parametric network alone are not sufficient. For example, on ScanObjectNN-C, the results of the Source-only and Non-parametric network-only are $59.35\%/59.45\%$, respectively, when using PointNet. By combining the logits from both Source-only and Non-parametric networks, our BFTT3D achieves better results at $54.46\%$. This demonstrates that the Non-parametric network provides complementary information to the source model.

**Comparison between static prototype memory and target-specific features.** We add static prototype memory to reduce the computational cost and memory consumption. We generate the target-specific logit by computing the similarity matrix. Hence, reducing the dimension of the feature memory matrix would accelerate the calculation. As shown in Table 3 of the paper, we could get similar performance only using $25\%$ of target features.

**Limitation.** It is empirically found that our method is sensitive to hyperparameter selection, particularly the wavelength $\alpha$, and the magnitude of trigonometric functions in raw embedding $\beta$. Overcoming this drawback could be a direction for future work.

## References


[1] 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. 1

[2] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016. 1

[3] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pages 6028–6039. PMLR, 2020. 1

[4] M Jehanzeb Mirza, Inkyu Shin, Wei Lin, Andreas Schriebl, Kunyang Sun, Jaesung Choe, Mateusz Kozinski, Horst Possegger, In So Kweon, Kuk-Jin Yoon, et al. Mate: Masked autoencoders are online 3d test-time learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16709–16718, 2023. 1, 2

[5] Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z Morley Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022. 2

[6] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. 1, 2