Point-Cache: Test-time Dynamic and Hierarchical Cache for Robust and Generalizable Point Cloud Analysis

Supplementary Material

6. Test-time Dynamic and Hierarchical Cache Construction and Adaptation

The overall pipeline of Point-Cache is described in Alg. 1. This pipeline consists of five steps, corresponding to the five modules illustrated in Fig. 2 of the main paper. Below, we explain several key operations in the algorithm.

Algorithm 1: Test-time Dynamic and Hierarchical Cache Construction and Adaptation

```
/* ----- 1. Input ----- */
Data: online test data, point cloud descriptions T,
      number of classes C, upper bound K,
      hyperparameters τ, α_g, β_g, α_l, β_l
Result: adapted class logits ŷ
while test sample Q do
    /* ----- 2. Encode ----- */
    e_q^g, e_q^l = f_p(Q);
    e_t = f_t(T);
    compute ŷ_zs = {ŷ_i|_{i=1}^C} using Eq. 1;
    obtain class L̂ = arg max_i {ŷ_i|_{i=1}^C} ;
    num = count(C_g, L̂) ;
    compute entropy h = −Σ_{i=1}^C ŷ_i log ŷ_i;
    /* ----- 3. Update ----- */
    if num < K then
        put (e_q^g, L̂, h) into global cache C_g;
        put (e_q^l, L̂) into local cache C_l;
    else
        h_max = retrieve(C_g, L̂);
        (e_p^{g,max}, L̂, h_max) = locate(C_g, L̂, h_max);
        (e_p^{l,max}, L̂) = locate(C_l, L̂, h_max);
        if h < h_max then
            (e_p^{g,max}, L̂, h_max) ← (e_q^g, L̂, h);
            (e_p^{l,max}, L̂) ← (e_q^l, L̂);
        end
    end
    /* ----- 4. Compute ----- */
    compute ŷ_g using Eq. 2;
    compute ŷ_l using Eq. 3;
    /* ----- 5. Adapt ----- */
    adapt ŷ_zs and obtain new logits ŷ using Eq. 4;
    return ŷ;
end
```

- The function 'count($\mathbf{C}_g, \hat{L}$)' calculates the number of cached fingerprints belonging to class $\hat{L}$ in the global cache $\mathbf{C}_g$.
- The function 'retrieve($\mathbf{C}_g, \hat{L}$)' returns the maximum entropy among the cached fingerprints of class $\hat{L}$ in $\mathbf{C}_g$.
- The function 'locate($\mathbf{C}_g$, $\hat{L}$, h_{max})' identifies the fingerprint with the highest entropy for class $\hat{L}$ in $\mathbf{C}_g$. Similarly, 'locate($\mathbf{C}_l$, $\hat{L}$, h_{max})' performs the same operation in the local cache $\mathbf{C}_l$.
- The operator $\leftarrow$ indicates that the fingerprint with the highest entropy is replaced by the fingerprint of the current sample Q if $h < h_{max}$.

7. Implementation Details

For the point encoder in ULIP [72] and ULIP-2 [73], we use PointBert [77] as the backbone. For the point encoder in OpenShape [29], we utilize the scaled version of PointBert (32.1M parameters), as detailed in Table 4 of the Appendix in the corresponding paper. For Uni3D, we employ the giant version, where the point encoder has 1,016.5M parameters. The pre-trained weights for these models are obtained from their public GitHub repositories. The zero-shot recognition accuracy (%) of the various large 3D models are the baselines for comparison.

Rather than relying on a single fixed template (*e.g.*, 'a point cloud object of a {class}') to describe a point cloud, we adopt 64 text templates to generate diverse descriptions of 3D objects, as in ULIP [72] and Point-PRC [50]. These descriptions are encoded into 64 text embeddings, which are then averaged to create a feature representation for a specific class.

8. Additional Results and Analysis

8.1. Test-time Robustness and Generalization

Robustness against data corruptions. We also create the corrupted versions for the three splits of ScanObjectNN according to the atomic operations in ModelNet-C [43] and conduct experiments on them. The results are reported in Tab. 5, 6 and 7. The proposed global and hierarchical cache models bring consistent and significant improvements across backbones, datasets and corruption types. For instance, +6.61% for ULIP on S-OBJ_ONLY-C, +6.05% for Uni3D on S-OBJ_BG-C, and +5.72% for OpenShape on S-PB_T50-_RS-C across 7 corruptions, compared to the corresponding zero-shot predictions. The results verify the ef-

fectiveness of Point-Cache in strengthening the robustness of large 3D model against data corruptions. Likewise as the observations from Tab. 1, the gains are not limited to corrupted data. Point-Cache also boosts the recognition accuracy of various models on original clean data.

Generalization from simulated to real data. We investigate the performances of Point-Cache on Sim-to-Real [19], which is used to evaluate the generalization from simulated data (in the source domain) to real data (in the target domain). Sim-to-Real introduces two evaluation settings: MN_11 $\rightarrow$ SONN_11 and SN_9 $\rightarrow$ SONN_9. MN is short for ModelNet, SN is short for ShapeNet and SONN is short for ScanObjectNN. SONN has three splits, as shown in Tab. 8. The results suggest our global cache model substantially raise the zero-shot accuracy of various large 3D models, *e.g.*, +3.77% based on Uni3D. Also, the hierarchical cache model leads the global one by a clear margin, *e.g.*, +3.46% based on ULIP-2 across 6 datasets, revealing the effectiveness of local cache again. Note that we also compare with prior strong baselines that are trained on the source domain, such as MetaSets [19], PDG [67] and I-OODG [86]. In contrast, Point-Cache is directly transferred to the target datasets of Sim-to-Real and totally training-free. As a result, we attain competitive or even better performances compared to those learning-based baselines.

8.2. Total Size of Full Hierarchical Cache

Here we explain how to calculate the total size of full hierarchical cache. The variables listed in Tab. 10 are vital to decide the total size of full hierarchical cache $\mathbf{C}_g \cup \mathbf{C}_l$, including the feature dimension d of $\mathbf{e}_p^g$ and $\mathbf{e}_p^l$, the upper bound K on the number of samples in each category, the number of parts m of a point cloud, and the number of classes C in the dataset.

Here we take (Uni3D, O-LVIS, Hierarchical Cache) as an example for computing the size of each item in the global and local cache.

- $\mathbf{E}_g$: 1156*3*512 = 1,775,616
- $\hat{\mathbf{L}}_g$: 1156*3 = 3,468
- $\mathbf{h}_g$: 1156*3 = 3,468
- $\mathbf{E}_l$: 1156*3*3*512 = 5,326,848
- $\hat{\mathbf{L}}_l$: 1156*3*3 = 10,404

So the total size of full hierarchical cache for (Uni3D, O-LVIS) is sum of these items, approximately 7.1M. We present the parameter counting for other backbones in Tab. 11. The results demonstrate the total size of a full hierarchical cache is very small, *e.g.*, 7.1M, particularly when compared to the hundreds of millions of parameters in a large multimodal 3D model, *e.g.*, 1016.5M in Uni3D. Therefore, Point-Cache introduces minimal additional computational and storage overhead, having little impact on memory usage and runtime efficiency, indicated by Tab. 3 and Tab. 4 of the main paper.

8.3. Memory Usage and Throughput

Memory. We compare the memory usage based on other large 3D models such as ULIP, ULIP-2 and OpenShape. In the experiments, a point cloud contains 1,024 points. The results are recorded in Tab. 12, 13 and 14. #Params count the total parameters in a large multimodal 3D model. We observe that our global and hierarchical cache model utilize same or slightly higher memory compared to the zero-shot baseline across backbones and datasets. For instance, OpenShape powered by our global cache consumes 7,058 MB GPU memory, same as the usage of zero-shot OpenShape. Moreover, with the number of 3D classes increasing rapidly, *e.g.*, $40 \rightarrow 216 \rightarrow 1,156$, the memory rises slowly, *e.g.*, $1,556 \rightarrow 1,558 \rightarrow 1,570$ for ULIP-2 with our hierarchical cache. The reason is same as we explained in the main paper: memory consumption is dominated by the numerous parameters of the large 3D model (*e.g.*, 32.3M #Params in the OpenShape point encoder alone) and the overhead of Point-Cache is ignorable.

Throughput. We test the throughput of Point-Cache on S-OBJ_ONLY and report the results in Tab. 16. The throughput is measured by the number of test samples per second (t/s) the model can process. Models with our global and hierarchical cache run slightly slower than zero-shot inference, *e.g.*, a 0.03 t/s drop for OpenShape with global cache and a 0.05 t/s drop for OpenShape with hierarchical cache, suggesting little computational overhead introduced by Point-Cache. In theory, the throughput is decided by the model itself and the GPU device used, instead of the dataset. In practice, the throughput on S-OBJ_ONLY is consistent with that on ModelNet40, as shown in Tab. 4 of the main paper.

8.4. Other Cache Models

Comparison with other cache models. There are only a few 3D point cloud cache models (Point-PEFT [46], BFTT3D [59] and Point-NN [73]). They have different pipelines and settings, making fair comparisons difficult, *e.g.*, (1) they use the entire training set (*with* real labels) to construct the cache offline, whereas Point-Cache constructs the cache using test data (*without* real labels) online; (2) they are not based on large 3D models and cannot recognize new classes in Omni3D and O-LVIS. In Tab. 17, we add comparisons with Point-NN (not a test-time method). The performance of Point-NN is expectedly better since it uses the real labels and the whole training set to build the cache.

9. Visualization

9.1. Relation with Previous Methods

Tab. 18 highlights the differences between existing cache models and Point-Cache. The proposed approach is a *dy-*

Table 5. **Comparison of recognition accuracy on S-OBJ_ONLY-C that includes 7 types of corruptions**. Results are reported for a corruption severity level of 2. Each clean point cloud contains 1024 points. The last column is the average across the 7 types of corruptions. SONN: ScanObjectNN.

Method	Original Data	Corruption Type							Avg.
	SONN	Add Global	Add Local	Drop Global	Drop Local	Rotate	Scale	Jitter	
ULIP [72]	49.05	31.50	34.77	51.29	38.38	48.36	44.58	36.83	40.82
+Global Cache(Ours)	**52.15**	**35.80**	37.01	54.39	41.82	49.74	45.09	**40.28**	43.45
+Hierarchical Cache(Ours)	**52.15**	32.01	**38.04**	**54.56**	**45.27**	**50.95**	**45.96**	39.24	**43.72**
ULIP-2 [73]	42.00	40.45	41.31	37.69	30.29	38.21	44.45	22.89	36.47
+Global Cache(Ours)	48.19	**49.05**	**46.30**	45.09	37.18	41.65	44.41	**25.99**	41.38
+Hierarchical Cache(Ours)	**51.98**	**49.05**	**46.30**	**48.88**	**40.45**	**45.78**	**45.09**	**25.99**	**43.08**
O-Shape [29]	53.18	49.91	46.30	52.15	36.66	46.64	46.82	30.81	44.18
+Global Cache(Ours)	56.80	56.45	51.98	54.56	40.45	**51.81**	**49.23**	37.69	48.88
+Hierarchical Cache(Ours)	**58.69**	**59.04**	**53.01**	**55.94**	**41.82**	51.12	48.54	**39.41**	**49.84**
Uni3D [88]	65.58	62.65	56.45	60.07	49.40	61.62	56.11	43.55	55.69
+Global Cache(Ours)	70.05	65.06	**59.38**	63.68	54.39	**63.34**	60.07	51.29	59.60
+Hierarchical Cache(Ours)	**70.22**	**65.40**	58.00	**64.20**	**54.91**	61.96	**62.13**	**53.18**	**59.97**

Table 6. **Comparison of recognition accuracy on S-OBJ_BG-C that includes 7 types of corruptions.** The results are reported for a corruption severity level of 2. Each clean point cloud has 1024 points. The last column is the average across the 7 types of corruptions.

Method	Original Data	Corruption Type							Avg.
	SONN	Add Global	Add Local	Drop Global	Drop Local	Rotate	Scale	Jitter	
ULIP [72]	45.96	27.19	25.82	45.61	34.25	40.96	40.10	30.98	34.99
+Global Cache(Ours)	48.88	**30.46**	**30.46**	**49.05**	39.59	**44.92**	**42.17**	31.84	**38.36**
+Hierarchical Cache(Ours)	**49.74**	28.23	30.12	48.71	**40.45**	43.55	40.28	**34.42**	37.97
ULIP-2 [73]	48.19	40.62	38.90	39.24	32.36	41.14	42.86	21.17	36.61
+Global Cache(Ours)	52.50	48.19	45.09	46.82	39.07	46.64	48.02	**26.51**	42.91
+Hierarchical Cache(Ours)	**54.73**	**51.64**	**47.16**	**50.95**	**39.76**	**53.01**	**51.81**	22.72	**45.29**
O-Shape [29]	55.94	49.40	48.19	52.67	42.51	48.88	47.16	31.84	45.81
+Global Cache(Ours)	59.72	57.49	51.12	**59.72**	**48.71**	**56.11**	**54.22**	35.28	**51.81**
+Hierarchical Cache(Ours)	**62.65**	**58.00**	**51.64**	59.55	47.85	54.91	53.36	**36.49**	51.69
Uni3D [88]	60.24	58.00	52.32	51.64	44.23	58.00	51.81	39.24	50.75
+Global Cache(Ours)	**63.86**	**66.27**	**57.83**	56.11	**50.77**	**61.62**	56.11	44.23	56.13
+Hierarchical Cache(Ours)	62.82	64.72	57.14	**58.52**	50.43	60.93	**59.55**	**46.30**	**56.80**

namic and *hierarchical* cache model that is constructed *entirely based on test data* for test-time point cloud recognition.

9.2. Point Cloud Encoding

Fig. 9 illustrates the detailed process of point cloud encoding, corresponding to the 'Encode' component of Fig. 2 in the main paper. For an input point cloud $P \in \mathbb{R}^{N\times 3}$, we first perform *farthest point sampling* to obtain M key points. Next, we search for k *nearest neighbors* for each key point to form M local point patches, which are transformed by a lightweight neural network (*e.g.*, 2-layer MLP in ViPFormer [49] or mini-PointNet [39] in PointBert [77]), as shown in Fig. 9 (a). Subsequently, a class token, along with the flattened point patches, is fed into the Transformer-based point encoder, generating the global feature $\mathbf{e}_p^g \in \mathbb{R}^d$ and local-part features. To save memory and computation, these local-part features (*e.g.* 512 in ULIP-2[73]) into m (*e.g.*, 5) centers using K-Means, resulting in $\mathbf{e}_p^l \in \mathbb{R}^{m\times d}$, as depicted in Fig. 9 (b).

9.3. The Global and Local Cache

Fig. 10 visualizes the global cache $\mathbf{C}_g$ and the local cache $\mathbf{C}_l$. $\mathbf{C}_g$ stores up to K global fingerprints $(\mathbf{e}_p^g, \hat{L}, h)$ per class from online test samples, while $\mathbf{C}_l$ records the local fingerprints $(\mathbf{e}_p^l, \hat{L})$ of corresponding samples. The global and local caches are empty at the beginning and then accept the fingerprints of online test samples. Both $\mathbf{C}_g$ and $\mathbf{C}_l$ are dynamically managed to prioritize high-quality samples, as outlined in Alg. 1. Note that the global and local caches are not necessarily full. This hierarchical design and the selective mechanism enable the creation of an more accurate profile for test data than previous cache methods, facilitating robust and generalizable point cloud analysis at test time.

Table 7. **Comparison of corruption generalization on S-PB_T50-_RS-C**, which is the hardest split of ScanObjectNN is used. Each clean point cloud is represented by 1024 points. SONN is short for ScanObjectNN.

Method	Original Data	Corruption Type							Avg.
	SONN	Add Global	Add Local	Drop Global	Drop Local	Rotate	Scale	Jitter	
ULIP [72]	29.29	19.26	18.39	30.99	23.91	27.48	26.34	21.44	23.97
+**Global Cache**(Ours)	32.37	22.87	20.85	33.31	27.90	30.85	**28.63**	24.53	26.99
+**Hierarchical Cache**(Ours)	**32.48**	**23.46**	**22.69**	**34.70**	**31.75**	**33.00**	28.28	**25.05**	**28.42**
ULIP-2 [73]	33.38	30.29	29.42	28.24	24.91	28.56	30.22	12.98	26.37
+**Global Cache**(Ours)	40.28	36.40	33.80	35.39	30.88	33.66	35.01	18.36	31.93
+**Hierarchical Cache**(Ours)	**42.40**	**35.70**	**34.42**	**37.75**	**34.21**	**36.26**	**36.09**	**19.12**	**33.36**
O-Shape [29]	41.12	32.41	35.60	37.80	27.34	36.61	35.22	18.88	31.98
+**Global Cache**(Ours)	42.16	40.32	37.58	42.02	33.76	41.53	**38.24**	24.12	36.80
+**Hierarchical Cache**(Ours)	**43.72**	**40.91**	**39.24**	**43.03**	**35.22**	**43.06**	37.40	**25.05**	**37.70**
Uni3D [88]	46.04	48.23	37.99	36.75	31.47	44.00	37.37	28.66	37.38
+**Global Cache**(Ours)	50.28	**52.57**	**42.23**	42.61	36.29	47.22	39.83	33.48	42.03
+**Hierarchical Cache**(Ours)	**51.13**	51.67	41.88	**44.59**	**38.79**	**49.03**	**41.05**	**34.70**	**43.10**

Table 8. **Comparison of recognition accuracy on Sim-to-Real**. Two evaluation settings are considered: MN_11 → SONN_11 and SN_9 → SONN_9. The dataset on the left side of → stands for simulated data, while the dataset on the right side indicates real-world data. 11 classes are shared between MN_11 and SONN_11, while 9 classes are common between SN_9 and SONN_9. The last column shows the average accuracy across 6 datasets. In the experiments, each point cloud is represented by 2,048 points. MN: ModelNet, SN: ShapeNet, -P: PointNet, -D: DGCNN. Note that our methods are **training-free** while prior methods (*e.g.*, PDG, MetaSets) **use the full training set** to build their models.

Method	Training?	MN_11 → SONN_11			SN_9 → SONN_9			Avg.
		OBJ	OBJ_BG	PB_T50_RS	OBJ	OBJ_BG	PB_T50_RS	
MetaSets-P [19]	✓	60.3	52.4	47.4	51.8	44.3	45.6	50.3
MetaSets-D [19]	✓	58.4	59.3	48.3	49.8	47.4	42.7	51.0
PDG-P [67]	✓	67.6	58.5	56.6	57.3	51.3	51.3	57.1
PDG-D [67]	✓	65.3	65.4	55.2	59.1	59.3	51.0	59.2
I-OODG [86]	✓	-	69.8	-	-	59.8	-	64.8
ULIP [72]	✗	57.05	50.32	32.60	61.00	61.00	44.38	51.06
+**Global Cache**(Ours)	✗	62.32	52.63	34.97	65.50	62.50	47.36	54.21
+**Hierarchical Cache**(Ours)	✗	**64.42**	**56.63**	**35.77**	**67.25**	**64.50**	**47.61**	**56.03**
ULIP-2 [73]	✗	52.42	53.89	41.57	51.50	59.25	46.35	50.83
+**Global Cache**(Ours)	✗	57.05	59.37	47.38	56.75	65.75	50.68	56.16
+**Hierarchical Cache**(Ours)	✗	**59.16**	**60.84**	**49.87**	**61.75**	**71.00**	**55.11**	**59.62**
OpenShape [29]	✗	62.32	64.42	48.52	64.00	70.25	53.55	60.51
+**Global Cache**(Ours)	✗	65.68	69.05	49.36	**71.00**	71.50	55.67	63.71
+ **Hierarchical Cache**	✗	**66.53**	**70.74**	**50.59**	71.50	71.00	**56.57**	**64.49**
Uni3D [88]	✗	72.63	74.53	55.76	75.50	77.00	57.98	68.90
+**Global Cache**(Ours)	✗	**76.21**	**77.26**	**59.10**	80.00	81.00	62.47	72.67
+**Hierarchical Cache**(Ours)	✗	74.11	76.00	57.92	**83.00**	**81.50**	**63.98**	**72.75**

9.4. Qualitative Analysis

We provide additional qualitative examples to demonstrate the step-by-step adaptation process of various large 3D models with Point-Cache, exhibited in Fig. 11, 12, 13 and 14. The results confirm that Point-Cache effectively assists large 3D models in correcting erroneous zero-shot predictions, reducing the classification entropy, and improving the recognition accuracy at test time. Notably, there are cases where the global cache model fails to adapt zero-shot predictions. For example, in the third row of Fig. 12, although ULIP-2+**GC** reduces the class probability for 'sink' from 73% to 53%, it still identifies 'sink' as the top-1 class. In contrast, ULIP-2+**HC** makes a sharp adjustment to the logits of ULIP-2+**GC** after incorporating local-part knowledge, promoting 'toilet' to the top-1 class (from 41% to 90%) and achieving a successful correction. Similar adaptations are observed in Fig. 11 (1st, 2nd, 4th, and 5th rows), Fig. 13 (1st, 4th, and 5th rows), and Fig. 14 (1st, 2nd, 3rd, and 6th rows), suggesting that the coarse-to-fine cache design is highly effective in capturing subtle differences among point cloud objects.

Table 9. **Comparison of recognition accuracy across a suite of datasets (no_lvis weights)**. S-PB_RS_T50 is the hardest split of ScanObjectNN. O-LVIS: Objaverse-LVIS. Omni3D: OmniObject3D. In Omni3D, each point cloud can be represented by a different number of points (pts). Note that Omni3D has 216 classes and O-LVIS has 1,156 classes. The last column is the average accuracy on these datasets.

Method	ModelNet40	S-PB_RS_T50	O-LVIS	Omni3D			**Avg.**
				1024 pts	4096 pts	16384 pts	
O-Shape [29]	85.05	54.01	**47.17**	33.64	34.16	34.25	48.05
+Global Cache(Ours)	**85.74**	**57.06**	47.06	37.11	**38.53**	**38.07**	**50.60**
+Hierarchical Cache(Ours)	85.70	56.40	45.69	**37.46**	38.36	38.05	50.28
Uni3D [88]	87.07	66.37	47.24	30.08	38.10	38.04	51.15
+Global Cache(Ours)	**87.93**	**68.58**	**47.51**	33.23	**39.51**	40.27	**52.84**
+Hierarchical Cache(Ours)	87.84	67.96	46.81	**33.91**	39.49	**40.49**	52.75

Table 10. **Statistics of feature dimension d, number of shots K per class, number of parts m per point cloud, and number of classes C in the dataset**. The used dataset is O-LVIS.

Backbone	d dims	K shots	m parts	C classes
ULIP	512	3	3	1,156
ULIP-2	512	3	3	1,156
OpenShape	1,280	3	3	1,156
Uni3D	512	3	3	1,156

Table 11. **Parameter count for the full hierarchical cache on O-LVIS, which covers 1,156 classes**. Capital letters in brackets indicate units of measurement.

Backbone	Global Cache			Local Cache		Total
	$\mathbf{E}_g$, (K)	$\hat{\mathbf{L}}_g$, (K)	$\mathbf{h}_g$ (K)	$\mathbf{E}_l$, (K)	$\hat{\mathbf{L}}_l$ (K)	Size (M)
ULIP	1775.6	3.5	3.5	5326.8	10.4	7.1
ULIP-2	1775.6	3.5	3.5	5326.8	10.4	7.1
O-Shape	4439.0	3.5	3.5	13,317.1	10.4	17.8
Uni3D	1775.6	3.5	3.5	5326.8	10.4	7.1

Table 12. **Comparison of memory usage (MB) based on ULIP.** The batch size is set to 1, and the experiments are conducted on an RTX 4090. The number below each dataset name indicates #Classes.

Method	ModelNet-C (40)	Omni3D (216)	O-LVIS (1,156)	#Params
ULIP	**1,556**	**1,558**	**1,556**	85.7M
+Global(Ours)	**1,556**	**1,558**	1,560	85.7M
+Hierar(Ours)	**1,556**	**1,558**	1,566	85.7M

Table 13. **Comparison of memory usage (MB) based on ULIP-2.** The batch size is set to 1, and the experiments are conducted on an RTX 4090. The number below each dataset name indicates #Classes.

Method	ModelNet-C (40)	Omni3D (216)	O-LVIS (1,156)	#Params
ULIP-2	**1,556**	**1,558**	**1,556**	85.7M
+Global(Ours)	**1,556**	**1,558**	1,560	85.7M
+Hierar(Ours)	**1,556**	**1,558**	1,570	85.7M

Table 14. **Comparison of memory usage (MB) based on OpenShape.** The batch size is set to 1, and the experiments are conducted on an RTX 4090. The number below each dataset name indicates #Classes.

Method	ModelNet-C (40)	Omni3D (216)	O-LVIS (1,156)	#Params
OpenShape	**7,056**	**7,058**	**7,116**	2,571.9M
+Global(Ours)	**7,056**	**7,058**	7,126	2,571.9M
+Hierar(Ours)	7,058	7,062	7,150	2,571.9M

Table 15. **Comparison of memory usage (MB) based on Uni3D.** The batch size is set to 1, and the experiments are conducted on an RTX 4090. The number below each dataset name indicates #Classes.

Method	ModelNet-C (40)	Omni3D (216)	O-LVIS (1,156)	#Params
Uni3D	**5,062**	**5,062**	**5,062**	1711.7M
+Global(Ours)	**5,062**	5,064	5,070	1711.7M
+Hierar(Ours)	5,064	5,068	5,090	1711.7M

Table 16. **Comparison of running throughput (t/s) for different models on S-OBJ_ONLY.** Each point cloud contains 1024 points. The batch size is set to 1 and the used device is an RTX 4090. The results are averaged over all test samples.

Method	Zero-shot	**+Global**(Ours)	**+Hierar**(Ours)
ULIP	**11.19**	11.16	11.14
ULIP-2	**11.19**	11.15	11.14
OpenShape	**9.86**	9.83	9.81
Uni3D	**9.62**	9.59	9.58

Table 17. **Comparison with Point-NN**. The comparison is unfair since Point-NN uses the training set to construct an offline cache.

Model	ModelNet (1,024 points)	ScanObjectNN (1,024 points)		
		OBJ_ONLY	OBJ_BG	PB_T50_RS
Point-NN	81.65	72.46	71.26	62.80
Uni3D	81.81	65.58	60.24	46.04
+Global	83.14	70.05	**63.86**	50.28
+Hierar	**83.87**	**70.22**	62.82	**51.53**

Table 18. **Comparison with other cache models**. In the first row, we select several key attributes of the cache models for comparison. 'Test-time' means whether the model is developed for test-time adaptation. 'T-set' indicates whether the cache model is solely built on the test set. 'Dynamic' and 'Hierarchical' represent whether the cache is dynamically managed and designed in a coarse-to-fine manner, respectively.

Cache Model	Test-time	T-set only	Dynamic	Hierarchical
Point-NN [84]	✗	✗	✗	✗
Point-PEFT [53]	✗	✗	✗	✗
BFTT3D [66]	✓	✗	✗	✗
Point-Cache	✓	✓	✓	✓

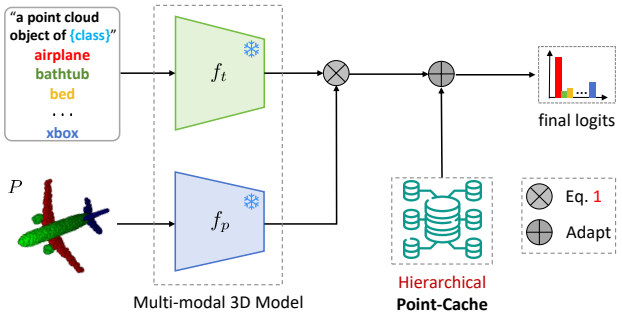


Figure 8. **Illustration of Point-Cache as a plug-and-play module**. We record global and local-part 3D features of high-quality test samples in the hierarchical cache. Then the hierarchical cache can be employed to adapt the zero-shot predictions of various large multi-modal 3D models.

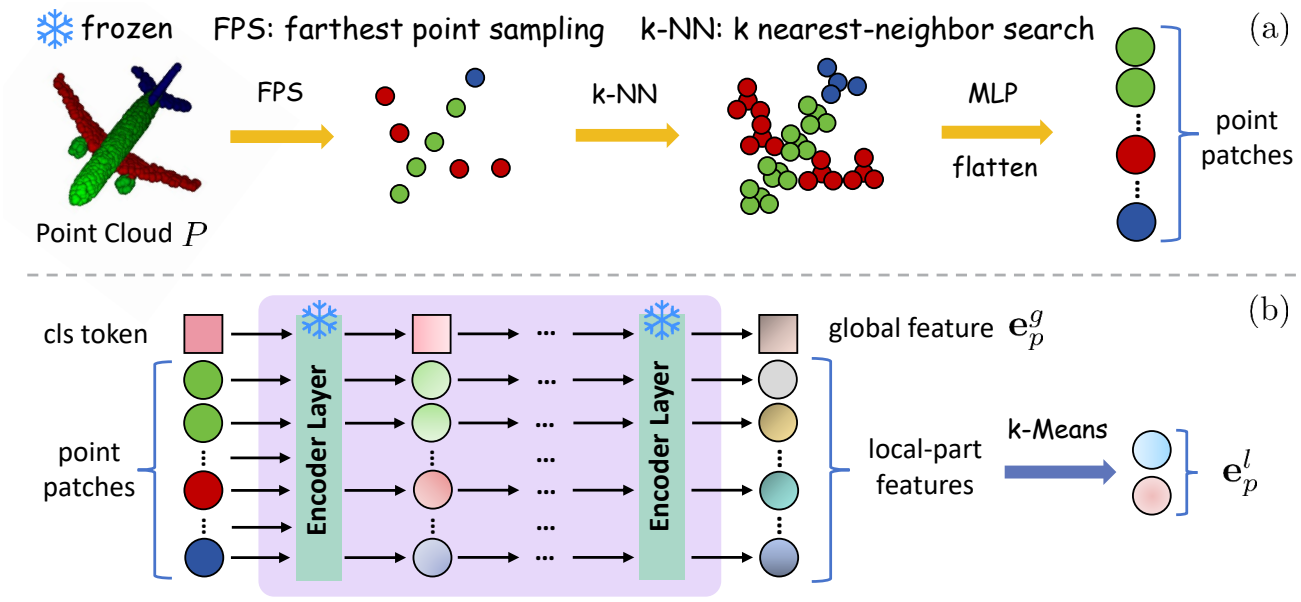


Figure 9. **Visualization of point cloud encoding**. Subfigure (a) illustrates the process of producing point patches, while subfigure (b) explains how the global feature and local-part features are generated.

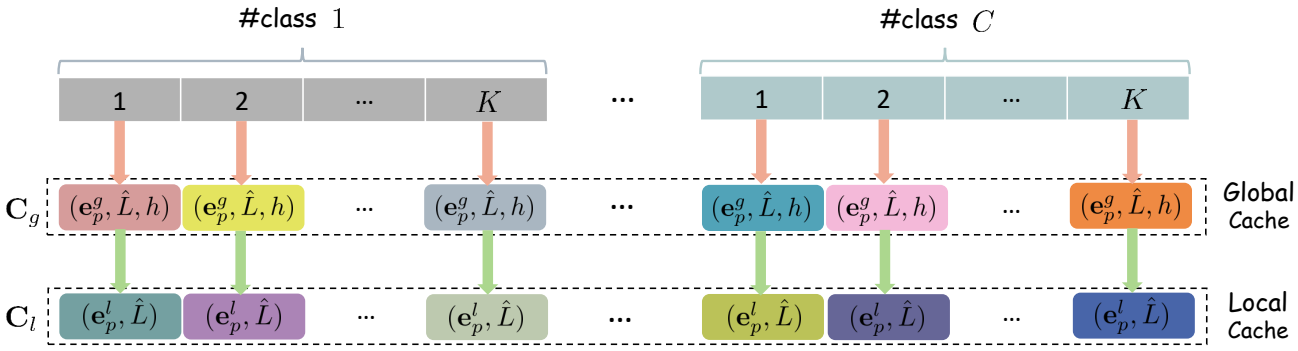
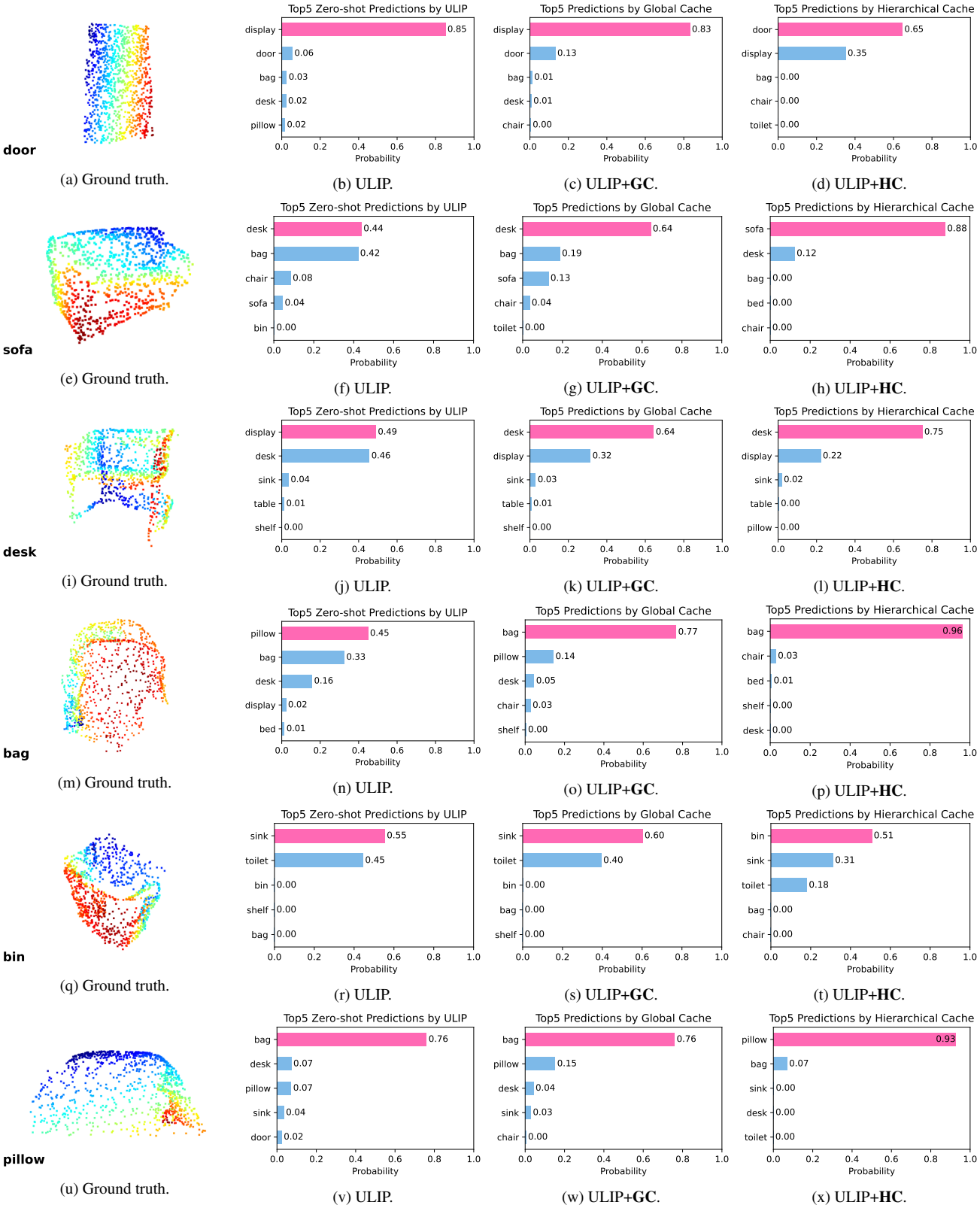


Figure 10. **Simulation of the global cache and local cache.** An upper bound K is set for the number of samples per class in the cache. The local fingerprint $(\mathbf{e}_p^l, \hat{L})$ of a sample P is stored in the local cache only if its global fingerprint $(\mathbf{e}_p^g, \hat{L}, h)$ qualifies for inclusion in the global cache. The global and local caches are initially empty and then updated according to Alg. 1. However, the global and local caches are not necessarily full. The full status of Point-Cache is determined by storing K global and local fingerprints in each of C categories.



(a) Ground truth. (b) ULIP. (c) ULIP+**GC**. (d) ULIP+**HC**.

(e) Ground truth. (f) ULIP. (g) ULIP+**GC**. (h) ULIP+**HC**.

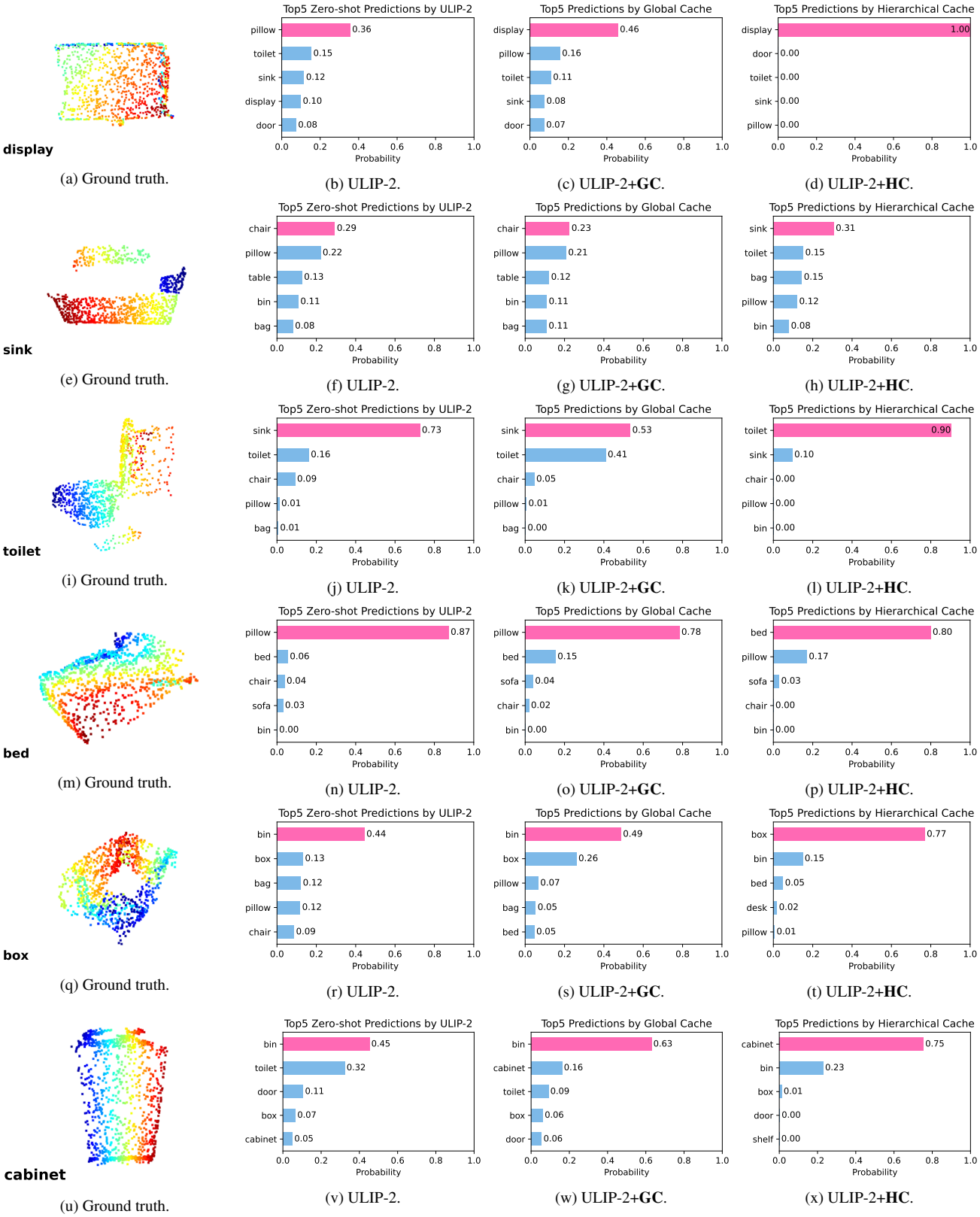
(i) Ground truth. (j) ULIP. (k) ULIP+**GC**. (l) ULIP+**HC**.

(m) Ground truth. (n) ULIP. (o) ULIP+**GC**. (p) ULIP+**HC**.

(q) Ground truth. (r) ULIP. (s) ULIP+**GC**. (t) ULIP+**HC**.

(u) Ground truth. (v) ULIP. (w) ULIP+**GC**. (x) ULIP+**HC**.

Figure 11. **ULIP zero-shot predictions before and after adaptation by Point-Cache.** The used dataset S-OBJ_ONLY-C (rotate, severity=2). Each 3D object contains 1,024 points. **GC**: global cache. **HC**: hierarchical cache.



(a) Ground truth. (b) ULIP-2. (c) ULIP-2+**GC**. (d) ULIP-2+**HC**.

(e) Ground truth. (f) ULIP-2. (g) ULIP-2+**GC**. (h) ULIP-2+**HC**.

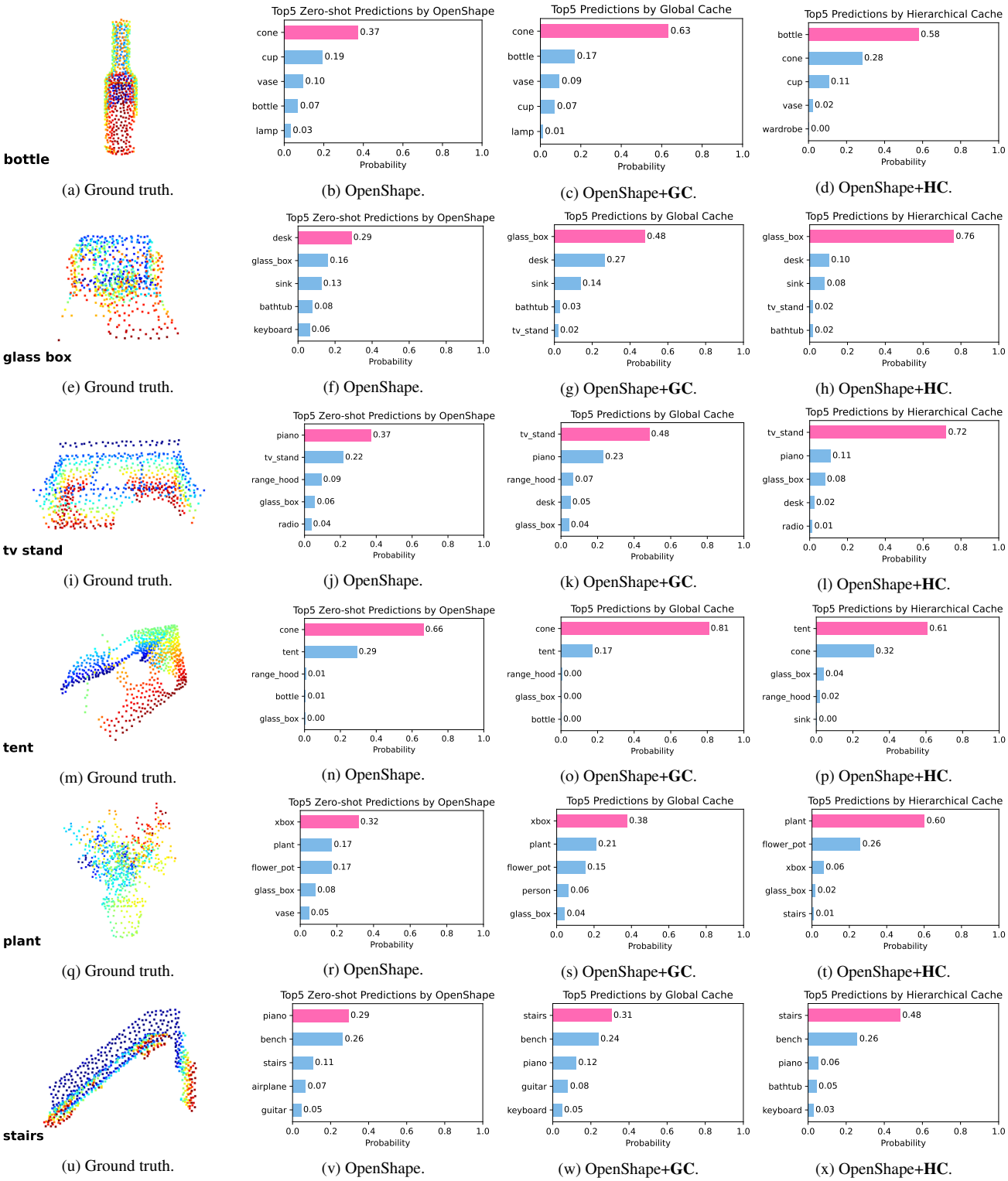
(i) Ground truth. (j) ULIP-2. (k) ULIP-2+**GC**. (l) ULIP-2+**HC**.

(m) Ground truth. (n) ULIP-2. (o) ULIP-2+**GC**. (p) ULIP-2+**HC**.

(q) Ground truth. (r) ULIP-2. (s) ULIP-2+**GC**. (t) ULIP-2+**HC**.

(u) Ground truth. (v) ULIP-2. (w) ULIP-2+**GC**. (x) ULIP-2+**HC**.

Figure 12. **ULIP-2 zero-shot predictions before and after adaptation by Point-Cache.** The used dataset is S-PB_T50_RS. Each 3D object contains 1,024 points. **GC**: global cache. **HC**: hierarchical cache.



(a) Ground truth. (b) OpenShape. (c) OpenShape+**GC**. (d) OpenShape+**HC**.

(e) Ground truth. (f) OpenShape. (g) OpenShape+**GC**. (h) OpenShape+**HC**.

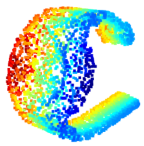
(i) Ground truth. (j) OpenShape. (k) OpenShape+**GC**. (l) OpenShape+**HC**.

(m) Ground truth. (n) OpenShape. (o) OpenShape+**GC**. (p) OpenShape+**HC**.

(q) Ground truth. (r) OpenShape. (s) OpenShape+**GC**. (t) OpenShape+**HC**.

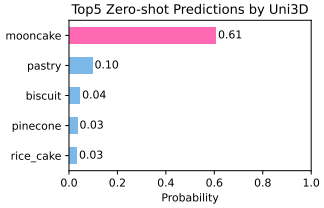
(u) Ground truth. (v) OpenShape. (w) OpenShape+**GC**. (x) OpenShape+**HC**.

Figure 13. **OpenShape zero-shot predictions before and after adaptation by Point-Cache.** The used dataset is ModelNet-C (drop_local, severity=2). Each 3D object contains 1,024 points. **GC**: global cache. **HC**: hierarchical cache.

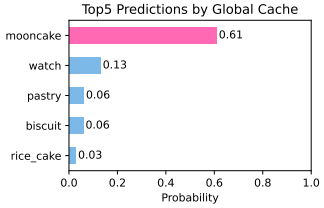


watch

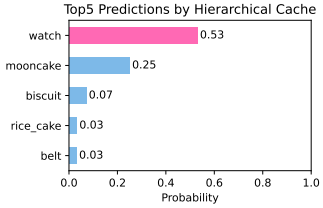
(a) Ground truth.



(b) Uni3D.



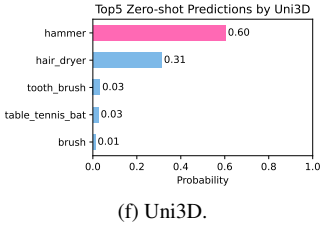
(c) Uni3D+**GC**.



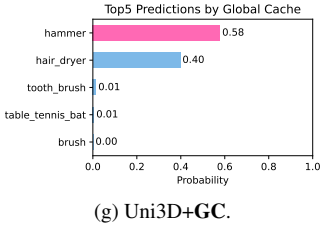
(d) Uni3D+**HC**.

hair dryer

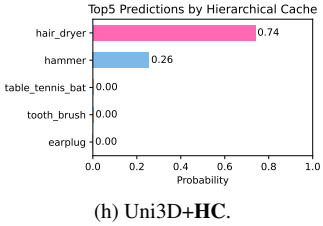
(e) Ground truth.



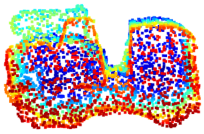
(f) Uni3D.



(g) Uni3D+**GC**.

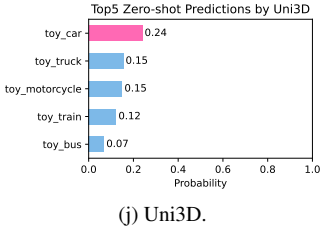


(h) Uni3D+**HC**.

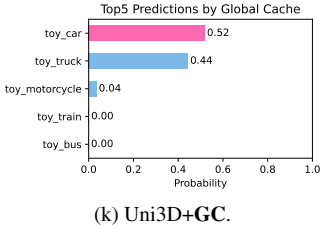


toy truck

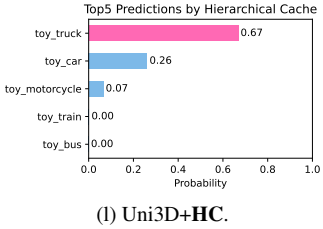
(i) Ground truth.



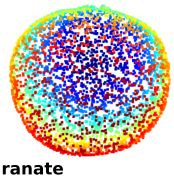
(j) Uni3D.



(k) Uni3D+**GC**.

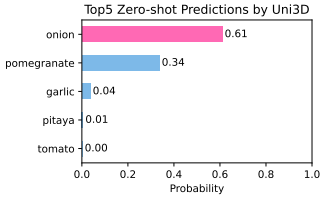


(l) Uni3D+**HC**.

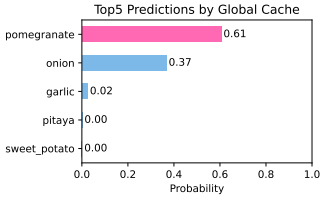


pomegranate

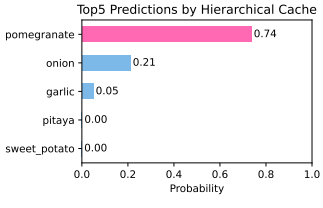
(m) Ground truth.



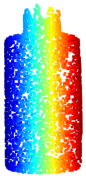
(n) Uni3D.



(o) Uni3D+**GC**.

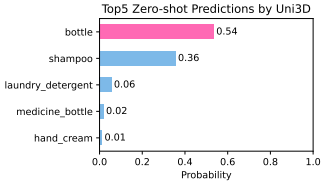


(p) Uni3D+**HC**.

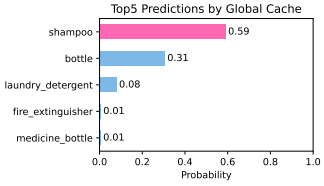


shampoo

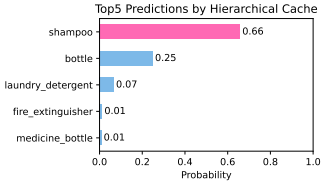
(q) Ground truth.



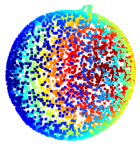
(r) Uni3D.



(s) Uni3D+**GC**.

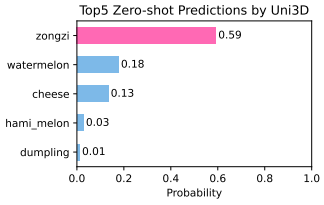


(t) Uni3D+**HC**.

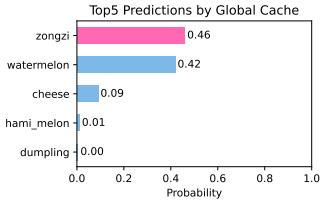


watermelon

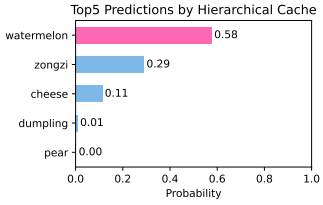
(u) Ground truth.



(v) Uni3D.



(w) Uni3D+**GC**.



(x) Uni3D+**HC**.

Figure 14. **Uni3D zero-shot predictions before and after adaptation by Point-Cache.** The used dataset is Omni3D. Each 3D object contains 4,096 points. **GC**: global cache. **HC**: hierarchical cache.