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

Hongyu Sun[1,2] Qiuhong Ke[2] Ming Cheng[2] Yongcai Wang[1] Deying Li[1] Chenhui Gou[2] Jianfei Cai[2]

[1]Department of Computer Science, Renmin University of China, China [2]Department of Data Science & AI, Monash University, Australia

Motivation

- **Problem:** Existing methods falter when faced with unpredictable shifts or new classes during online inference. They're stuck in a "training-only" mindset—impractical for dynamic, real-world scenarios.
- **Objective:** Enter Point-Cache—a training-free, test-time solution! We built a dynamic, hierarchical cache that learns on the fly from online test data alone. It captures both global shapes and local details of point clouds, adapting robustly to anything thrown its way.

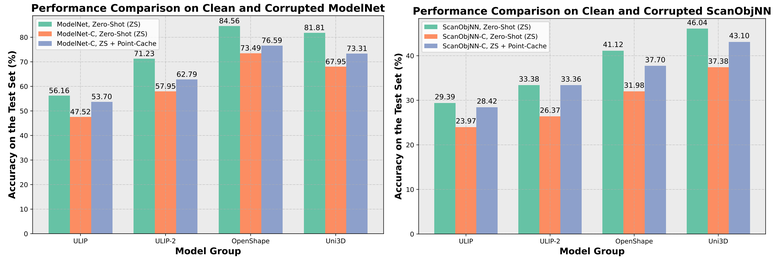


Figure 1. **Recognition accuracy comparison on clean and corrupted point cloud datasets.** The suffix -C indicates datasets with corruptions. Models experience a severe performance drop when data corruptions arise.

Highlights

- First to tackle test-time point cloud recognition with only test data, generalizing to known and unseen classes, in a training-free manner.
- A plug-and-play hierarchical (global + local) cache boosting large 3D models like ULIP & ULIP-2 & OpenShape & Uni3D.
- Massive gains: +6.18% on ModelNet-C, +6.99% on ScanObjNN-C (hardest), +2.10% on Objaverse-LVIS (1,156 classes) across 8 benchmarks.

Challenges

We can obtain the global feature of a point cloud P and its text encoding easily. However, it is *non-trivial* to represent the local-part features of the point cloud P due to the following challenges.

01 Part Feature Capture

- Difficult to define parts precisely and get their features for various 3D objects of different classes

02 Part Feature Storage

- Expensive to store multiple part features instead of a global feature for a single point cloud

Figure 2. **Challenges in encoding part feature** for 3D objects of different classes: Part Feature Capture & Storage.

Proposed Methodology

We propose Point-Cache, a cache model to store critical fingerprints (which are essentially key-value or feature-label pairs) of test samples, to fully exploit the data distribution during test time.

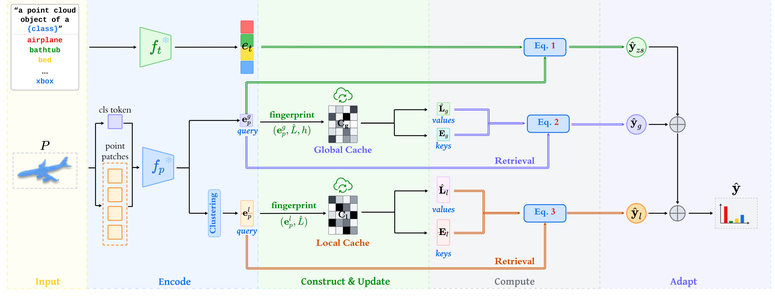


Figure 3. **The overall pipeline of Point-Cache.** The zero-shot predictions $\hat{\mathbf{y}}_{zs}$ of large 3D models are effectively adapted by our global cache logits $\hat{\mathbf{y}}_g$ and local cache logits $\hat{\mathbf{y}}_l$ to handle the distribution shifts and data corruptions.

Constuction & Update of Hierarchical Cache

Global cache: $\mathbf{C}_g = (\mathbf{E}_g, \hat{\mathbf{L}}_g, \mathbf{h}_g)$, storing (*feature*, *label*, *entropy*) triplets of online test samples.
Local cache: $\mathbf{C}_l = (\mathbf{E}_l, \hat{\mathbf{L}}_l)$, storing (*feature*, *label*) pairs of online test samples.

Construction & Update of **Global Cache** and **Local Cache**:

- K: the **maximum** number of samples per class in the cache. Initially, both $\mathbf{C}_g$ and $\mathbf{C}_l$ are empty.
- When a new test sample P arrives, we generate its **global** fingerprint $(\mathbf{e}_p^g, \hat{L}, h)$ and **local** fingerprint $(\mathbf{e}_p^l, \hat{L})$, and then assign it to class $\hat{L}$.
- Append the global fingerprint $(\mathbf{e}_p^g, \hat{L}, h)$ and the local fingerprint $(\mathbf{e}_p^l, \hat{L})$ of current sample to class $\hat{L}$ **if** the number of stored samples for class $\hat{L}$ is **less than** K.
- Otherwise, **select** the fingerprint with **highest entropy** within class $\hat{L}$, **replace** this fingerprint with current fingerprint if the entropy of current sample is lower.

Test-time Adaptation by Our Hierarchical Cache

Zero-shot prediction of large 3D models

$$\hat{\mathbf{y}}_{zs} = \{\hat{y}_i|_{i=1}^{C}\}, \quad \hat{y}_i = \frac{\exp(sim(\mathbf{e}_p^g, \mathbf{e}_t^i)/\tau)}{\sum_{j=1}^{C}\exp(sim(\mathbf{e}_p^g, \mathbf{e}_t^j)/\tau)} \tag{1}$$

Adaptation by global cache

$$A_g = \exp(-\beta_g(1-\mathbf{e}_q^g\mathbf{E}_g^\top)),\ \hat{\mathbf{L}}_g = \mathrm{OH}(\hat{\mathbf{L}}_g),\ \hat{\mathbf{y}}_g = A_g\hat{\mathbf{L}}_g \tag{2}$$

Adaptation by local cache

$$A_l = \exp(-\beta_l(1-\mathbf{e}_q^l\mathbf{E}_l^\top)),\ \hat{\mathbf{L}}_l = \mathrm{OH}(\hat{\mathbf{L}}_l),\ \hat{\mathbf{y}}_l = \phi(A_l\hat{\mathbf{L}}_l) \tag{3}$$

Overall prediction

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}_{zs} + \alpha_g\hat{\mathbf{y}}_g + \alpha_l\hat{\mathbf{y}}_l \tag{4}$$

Experiments

Robustness evaluation on ModelNet-C, which contains 7 types of data corruptions. Each clean point cloud has 1024 points and the corruption severity level is 2. The last column is the average.

Method	Clean Data	Corruption Type							Avg.
	ModelNet	Add Global	Add Local	Drop Global	Drop Local	Rotate	Scale	Jitter	
ULIP [72]	56.16	33.55	43.92	54.70	50.89	55.27	50.20	44.08	47.52
+**Global Cache**(Ours)	62.12	45.79	**47.98**	56.85	53.89	60.25	54.34	48.91	52.56
+**Hierarchical Cache**(Ours)	**64.22**	**46.15**	47.85	**59.16**	**56.00**	**61.47**	**55.35**	**49.92**	**53.70**
ULIP-2 [73]	71.23	65.15	54.62	68.76	57.98	70.30	67.10	21.76	57.95
+**Global Cache**(Ours)	73.95	67.02	59.32	71.35	61.59	72.37	68.40	28.20	61.18
+**Hierarchical Cache**(Ours)	**74.53**	**68.11**	**61.26**	**73.22**	**63.65**	**73.34**	**70.42**	**29.50**	**62.79**
O-Shape [29]	**84.56**	71.64	67.79	81.56	73.58	82.01	78.48	59.36	73.49
+**Global Cache**(Ours)	84.52	74.72	72.77	**82.41**	75.12	**83.18**	**78.93**	67.91	76.43
+**Hierarchical Cache**(Ours)	84.04	**74.84**	**73.70**	82.21	**76.26**	82.66	78.12	**68.35**	**76.59**
Uni3D [88]	81.81	72.45	56.36	68.15	67.18	79.94	75.36	56.24	67.95
+**Global Cache**(Ours)	83.14	76.13	66.49	71.43	69.81	81.52	75.85	61.43	71.81
+**Hierarchical Cache**(Ours)	**83.87**	**77.51**	**71.15**	**72.16**	**70.75**	**81.77**	**77.31**	**62.52**	**73.31**

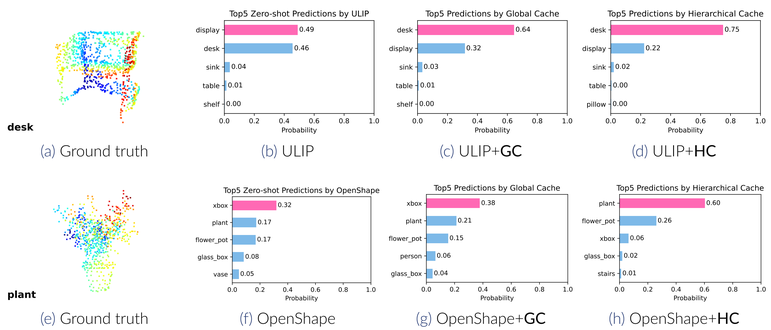
Figure 4. **Comparison of recognition accuracy on ModelNet-C** that contains 7 types of corruptions.

Generalization evaluation on a suite of datasets, where Omni3D and O-LVIS include 216 and 1,156 classes, respectively. The numbers in the bracket are the number of points per 3D object.

Method	ModelNet40 (10000 pts)	S-PB_RS_T50 (2048 pts)	O-LVIS (10000 pts)	Omni3D (1024 pts	4096 pts	16384 pts)	Avg.
ULIP [72]	58.75	46.44	6.24	8.39	7.75	7.28	22.48
+**Global Cache**(Ours)	61.22	50.21	**7.02**	10.00	9.36	8.43	24.37
+**Hierarchical Cache**(Ours)	**62.93**	**51.80**	**7.02**	**10.47**	**9.75**	**8.90**	**25.15**
ULIP-2 [73]	72.97	47.13	30.26	26.36	29.20	26.58	38.75
+**Global Cache**(Ours)	74.51	51.70	**32.65**	28.51	31.10	28.53	41.17
+**Hierarchical Cache**(Ours)	**75.53**	**54.98**	32.36	**29.37**	**31.24**	**29.44**	**42.15**
Uni3D [88]	88.41	65.19	**55.42**	31.52	41.98	41.86	54.09
+**Global Cache**(Ours)	88.86	**68.51**	53.36	34.97	45.13	45.19	56.00
+**Hierarchical Cache**(Ours)	**89.18**	68.24	55.19	**35.82**	**45.60**	**45.89**	**56.65**

Figure 5. **Comparison of recognition accuracy across a suite of datasets.** S-PB_RS_T50 is the hardest split of ScanObjectNN. O-LVIS: Objaverse-LVIS. Omni3D: OmniObject3D, where objects have a different number of points.

Visualization of step-by-step adaptation for large 3D models by our Global Cache (GC) and Hierarchical cache (HC). The hierarchical cache model consists of a global and a local cache.



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

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