

465 Appendix

466 This appendix provides detailed supplementary material to support the main paper, *Boosting Skeleton-*
 467 *based Zero-Shot Action Recognition with Training-Free Test-Time Adaptation*. It is organized into
 468 four sections: (1) **Algorithm** gives the full Skeleton-Cache pseudocode and clarifies notation. (2)
 469 **Implementation Details** covers the efficiency analysis, LLM prompt design, and the spatial/temporal
 470 partitioning strategy required for replication. (3) **More Quantitative Results** extends the empirical
 471 study with extra ZSL methods, comparisons with generic TTA baselines, and cache update with
 472 adapted logits. Finally, (4) **Analytical Visualizations** analyzes adaptation effects through weight
 473 heat-maps, confusion matrices, top-5 prediction changes, and per-class accuracy changes. These
 474 sections collectively enhance the understanding of Skeleton-Cache’s functionality, reproducibility,
 475 and performance.

476 A Algorithm

477 This section provides a formal description of the Skeleton-Cache algorithm through pseudocode and
 478 clarifies the notation used throughout our paper. The algorithm is designed to enhance the general-
 479 ization of skeleton-based zero-shot action recognition (SZAR) models by dynamically maintaining
 480 a non-parametric cache of confident exemplars and integrating similarity-based predictions with
 481 LLM-guided weights, all without requiring model retraining.

Algorithm 1 Skeleton-Cache: Training-Free Test-Time Adaptation for SZAR

Require: Test skeleton sequences, action class descriptions, number of unseen classes $|\mathcal{Y}_u|$, cache
 size K , spatial groups P , temporal segments Z , hyperparameters α_s, β

Ensure: Adapted class predictions \hat{y}

```

1: Initialize caches  $\mathcal{C}_j \leftarrow \emptyset$  for  $j = 1, \dots, |\mathcal{Y}_u|$ 
2: for all test samples do ▷ Phase 1: Feature Extraction
3:   Extract skeleton features  $F \in \mathbb{R}^{N \times T \times V}$  with  $\mathcal{M}$ 
4:    $g \leftarrow \frac{1}{V|T|} \sum_{t=1}^T \sum_{v=1}^V F_{:,t,v}$ 
5:   for  $p = 1$  to  $P$  do
6:      $s_p \leftarrow \frac{1}{|V_p|T} \sum_{v \in V_p} \sum_{t=1}^T F_{:,t,v}$ 
7:   for  $z = 1$  to  $Z$  do
8:      $t_z \leftarrow \frac{1}{V|T_z|} \sum_{t \in T_z} \sum_{v=1}^V F_{:,t,v}$ 
9:    $e_k \leftarrow \text{concat}(g, s_1, \dots, s_P, t_1, \dots, t_Z)$  ▷ Phase 2: Initial Prediction
10:  Obtain initial label  $\hat{y}$  and entropy  $\hat{h}$  from  $\mathcal{M}$  ▷ Phase 3: Cache Update
11:  if  $|\mathcal{C}_{\hat{y}}| < K$  then
12:     $\mathcal{C}_{\hat{y}} \leftarrow \mathcal{C}_{\hat{y}} \cup \{(e_k, \hat{h})\}$ 
13:  else
14:    Let  $(e^{\max}, h^{\max}) = \arg \max_{(e,h) \in \mathcal{C}_{\hat{y}}} h$ 
15:    if  $\hat{h} < h^{\max}$  then
16:      Replace  $(e^{\max}, h^{\max})$  with  $(e_k, \hat{h})$  ▷ Phase 4: Similarity Computation
17:  for  $j = 1$  to  $|\mathcal{Y}_u|$  do
18:    for all  $(k_{j,i}, h_{j,i}) \in \mathcal{C}_j$  do
19:       $a_{j,i}^{(d)} \leftarrow \exp[-\beta(1 - \cos(q^{(d)}, k_{j,i}^{(d)}))]$ 
20:     $o^{(d)} \leftarrow a^{(d)}Y$  for each descriptor  $d$  ▷ Phase 5: LLM-Guided Adaptation
21:    Generate normalized weight  $w \in \mathbb{R}^{P+Z+1}$  via LLM
22:     $s \leftarrow w \cdot \text{concat}(o^{(d)})$ 
23:     $\hat{y} \leftarrow \arg \max(\text{softmax}(\hat{\varphi} + \alpha_s s))$ 

```

482 **Notation.** We denote the original zero-shot logits produced by the frozen SZAR model as $\hat{\phi}$. The
 483 enhanced logits, augmented by Skeleton-Cache, are defined as $\varphi = \hat{\phi} + \alpha_s s$ (see Eq. 12 in the
 484 main paper). The entropy of a probability distribution is denoted by $\mathcal{H}(\cdot)$. The cache for class j
 485 is represented as \mathcal{C}_j , containing tuples of feature representations (cache keys e_k), predicted labels

(\mathbf{e}_y), and confidence scores (\mathbf{e}_h). The descriptors include global (\mathbf{g}), spatial (\mathbf{s}_p), and temporal (\mathbf{t}_z) components, concatenated to form the cache key \mathbf{e}_k . The pseudocode outlines the Skeleton-Cache method, detailing the feature extraction, cache update, similarity computation, and LLM-guided fusion processes.

B Implementation Details

This section provides comprehensive details on the implementation of Skeleton-Cache, including efficiency analysis, the LLM prompt template for weight extraction, and our spatial and temporal partitioning strategies. These details ensure reproducibility and clarify the computational overhead and practical considerations of our method.

B.1 Efficiency Analysis

As mentioned in Section 4.3, Table 5 presents the memory footprint and latency overhead of Skeleton-Cache across four zero-shot splits. The per-class memory consumption ranges from 120.0 KB to 128.0 KB, with the total memory requirement not exceeding 2.94 MB even for the largest split (NTU120 96/24). The computational latency varies from 0.40 seconds for the smallest split (NTU60 55/5) to 6.21 seconds for the largest (NTU120 96/24), reflecting the increased processing demands of cache operations as the number of unseen classes grows. These metrics confirm that Skeleton-Cache introduces minimal overhead compared to the base SZAR model, making it a lightweight solution suitable for real-time applications.

Table 5: Efficiency overhead of *Skeleton-Cache* on four zero-shot splits.

Split	Unseen classes	Memory		Extra
		per class (KB)	total (MB)	latency (s)
NTU60 55/5	5	128.0	0.64	0.40
NTU60 48/12	12	120.0	1.44	1.98
NTU120 110/10	10	128.0	1.28	2.27
NTU120 96/24	24	122.7	2.94	6.21

B.2 Prompt Template for LLM-Guided Weights

Skeleton-Cache Prompt (per action class)

You are an expert in human-action understanding. Given the action class `<ACTION>`, answer the following three questions **without adding commentary**.

- Spatial importance.** The human body is divided into four regions: [Head, Torso, Arms, Legs]. Provide a list of four non-negative numbers that sum to 1, corresponding to the relative importance of each region for recognising `<ACTION>`.
Format: "spatial": [w_head, w_torso, w_arms, w_legs]
- Temporal importance.** The action sequence is divided into three phases: [Beginning, Middle, End]. Provide a list of three non-negative numbers that sum to 1, indicating the relative importance of each phase.
Format: "temporal": [w_begin, w_mid, w_end]
- Global vs local preference.** Provide a single number $\gamma \in [0, 1]$ indicating how much the action should be recognised holistically ($\gamma \approx 1$) versus by local parts/phases ($\gamma \approx 0$).
Format: "gamma": γ

Return **one compact JSON object** with keys "spatial", "temporal", and "gamma". Do not include any other keys, text, or explanations.

Example Response (action = "Waving")

```
{
  "spatial": [0.05, 0.10, 0.70, 0.15],
  "temporal": [0.15, 0.60, 0.25],
  "gamma": 0.30
}
```

506

LLM-guided weight extraction workflow. To obtain the dataset-specific fusion weights, we employ a streamlined process leveraging LLM-based knowledge extraction. For each action class in the target dataset, we first construct a class-specific query by inserting the action name into our prompt template. These queries are then batch-processed through an LLM API (we use gpt-4-turbo with temperature $\tau = 0$), which returns structured JSON responses containing three key components: spatial importance weights $\mathbf{w}_{\text{spa}}^{(c)} \in \mathbb{R}^P$ across body regions, temporal importance weights $\mathbf{w}_{\text{tmp}}^{(c)} \in \mathbb{R}^Z$ across action phases, and a global-local preference parameter $\gamma^{(c)} \in [0, 1]$. We then process these raw outputs to construct a per-class weight vector $\tilde{\mathbf{w}}^{(c)} = [\gamma^{(c)}, (1 - \gamma^{(c)}) \cdot \mathbf{w}_{\text{spa}}^{(c)\top}, (1 - \gamma^{(c)}) \cdot \mathbf{w}_{\text{tmp}}^{(c)\top}]$, followed by ℓ_1 -normalisation to obtain the final vector $\mathbf{w}^{(c)}$. The complete LLM-prior matrix $\mathbf{W} \in \mathbb{R}^{|C| \times (P+Z+1)}$ is formed by stacking these normalized weight vectors for all classes and stored as a NumPy array for efficient access during test-time inference. This entire process is performed once per dataset without requiring any model training or fine-tuning, making it computationally efficient and broadly applicable across different skeleton-based action recognition tasks.

520 B.3 Spatial and Temporal Partitioning

This subsection describes the partitioning strategy for spatial and temporal descriptors used in Skeleton-Cache for the NTU RGB+D and PKU-MMD datasets. The spatial partitioning divides the skeleton joints into semantically meaningful groups, while the temporal partitioning employs a uniform segmentation approach to capture action dynamics.

Spatial Partitioning. For both NTU RGB+D and PKU-MMD datasets, we partition the skeleton joints into four spatial groups corresponding to distinct body regions: head, torso, arms and feet. These groups are defined based on the joint indices provided in the datasets, ensuring that each group captures anatomically relevant features for action recognition. The specific joint indices for each spatial group in the NTU RGB+D dataset are as follows:

- 530 • **Head:** Joints {2, 3, 4, 8, 20}, covering the head and upper neck regions, with additional
531 interpolation to include seven joints for robust feature extraction.
- 532 • **Torso:** Joints {0, 1, 4, 8, 12, 16, 20}, including the spine, shoulders, and central joints, with
533 interpolation to include five additional joints for comprehensive torso representation.
- 534 • **Arms:** Joints {4, 5, 6, 7, 8, 9, 10, 11, 21, 22, 23, 24}, encompassing both hands and wrists
535 to capture fine-grained hand movements critical for actions like “writing” or “waving.”
- 536 • **Feet:** Joints {0, 12, 13, 14, 15, 16, 17, 18, 19}, covering the lower body, including hips,
537 knees, and ankles, with interpolation to include three additional joints to enhance leg motion
538 capture.

For the PKU-MMD dataset, the same semantic partitioning is applied, mapping the corresponding joint indices to these four regions while accounting for any dataset-specific differences in joint definitions. The spatial features \mathbf{s}_p for each group $p \in \{1, 2, 3, 4\}$ are computed by averaging the feature representations $\mathbf{F}_{:,t,v}$ over the joints $v \in V_p$ and time steps $t \in [1, T]$, as described in Algorithm 1. This partitioning ensures that Skeleton-Cache captures localized motion cues relevant to specific body parts, enhancing the discriminative power of the descriptors.

Temporal Partitioning. The temporal partitioning strategy divides each skeleton sequence into three equal segments to capture the dynamics of the action across its progression: beginning, middle, and end phases. For a sequence with T frames, the temporal segments are defined as follows:

- 548 • **Beginning:** Frames $t \in [1, \lfloor T/3 \rfloor]$.

- **Middle:** Frames $t \in [\lfloor T/3 \rfloor + 1, \lfloor 2T/3 \rfloor]$.
- **End:** Frames $t \in [\lfloor 2T/3 \rfloor + 1, T]$.

For each segment $z \in \{1, 2, 3\}$, the temporal features \mathbf{t}_z are computed by averaging the feature representations $\mathbf{F}_{:,t,v}$ over the frames $t \in T_z$ and joints $v \in [1, V]$, as outlined in Algorithm 1. This uniform segmentation ensures that each phase of the action is equally represented, allowing Skeleton-Cache to model temporal evolution without requiring manual annotation of action phases. The approach is computationally efficient and generalizes well across datasets with varying sequence lengths, as demonstrated in our experiments on NTU RGB+D and PKU-MMD.

C More Quantitative Results

To further assess the robustness and generalizability of Skeleton-Cache, this section broadens the quantitative evaluation with additional zero-shot learning methods, comparisons against generic test-time adaptation baselines, and analysis of cache update strategies with adapted logits.

C.1 ZSL Performance with Additional Methods

While the main paper evaluates Skeleton-Cache on several established ZSL methods, we extend the analysis here by experimenting with four additional ZSL methods on the NTU RGB+D 60 and NTU RGB+D 120 datasets. These experiments reinforce the trend observed in the main paper: more advanced ZSL methods exhibit greater performance improvements when augmented with Skeleton-Cache. This is attributed to Skeleton-Cache’s reliance on high-confidence samples for populating the cache model, as advanced ZSL methods typically produce predictions with lower entropy compared to earlier methods, which often suffer from high entropy and class imbalance issues, such as misclassifying all samples of one class into another.

Table 6: Comparison of ZSL accuracy (%) on NTU RGB+D datasets. Our proposed Skeleton-Cache (SC) method shows greater improvements on more advanced ZSL methods due to their ability to generate more confident predictions for caching.

Method	NTU RGB+D 60		NTU RGB+D 120	
	55/5 Split	48/12 Split	110/10 Split	96/24 Split
ReViSE [9]	53.91	17.49	55.04	32.38
JPoSE [28]	64.82	28.75	51.93	32.44
CADA-VAE [19]	76.84	28.96	59.53	35.77
SynSE [8]	75.81	33.30	62.69	38.70
SMIE [32]	77.98	40.18	65.74	45.30
PURLS [33]	79.22	40.99	71.95	52.01
SA-DVAE [15]	82.37	41.38	68.77	46.12
STAR [3]	81.40	45.10	63.30	44.30
ReViSE+SC	54.70 ^{↑0.79}	18.05 ^{↑0.56}	55.83 ^{↑0.79}	33.01 ^{↑0.63}
JPoSE+SC	66.31 ^{↑1.49}	29.84 ^{↑1.09}	53.44 ^{↑1.51}	33.56 ^{↑1.12}
CADA-VAE+SC	78.94 ^{↑2.10}	30.49 ^{↑1.53}	61.53 ^{↑2.00}	37.30 ^{↑1.53}
SynSE+SC	79.73 ^{↑3.92}	38.82 ^{↑5.52}	68.47 ^{↑5.78}	44.10 ^{↑5.40}
SMIE+SC	82.63 ^{↑4.65}	44.17 ^{↑3.99}	72.98 ^{↑7.24}	50.44 ^{↑5.14}
PURLS+SC	85.46 ^{↑6.24}	45.22 ^{↑4.23}	77.60 ^{↑5.65}	56.83 ^{↑4.82}
STAR+SC	88.82 ^{↑7.42}	52.03 ^{↑6.93}	69.54 ^{↑6.24}	50.02 ^{↑5.72}
SA-DVAE+SC	89.41 ^{↑7.04}	47.83 ^{↑6.45}	74.29 ^{↑5.52}	53.14 ^{↑7.02}

Table 6 compares the Top-1 accuracy of eight ZSL methods, including four additional methods (ReViSE, JPoSE, CADA-VAE, and STAR) not evaluated in the main paper, on four zero-shot splits of NTU RGB+D 60 and NTU RGB+D 120. The results show that Skeleton-Cache (SC) consistently improves performance across all methods, with larger gains observed for more advanced methods like SA-DVAE and PURLS. For instance, SA-DVAE+SC achieves a 7.04% improvement on the NTU60 55/5 split, compared to only a 2.79% improvement for ReViSE+SC. This trend is consistent across splits, as advanced methods generate more confident predictions, enabling Skeleton-Cache to effectively cache representative exemplars. In contrast, earlier methods like ReViSE and JPoSE

often produce high-entropy predictions or collapse to predicting a single class, limiting the cache’s effectiveness.

C.2 Comparison with Generic TTA Baselines

Table 7 contrasts **Skeleton-Cache** with mainstream test-time adaptation (TTA) methods—gradient-based prompt-tuning (TPT [24], DiffTPT [6]), training-free prototype or attention modulation (AdaNPC [31], CALIP [7]) and dual-cache retrieval (TDA [11])—using a shared frozen *PURLS* backbone. By leveraging skeleton-specific priors, our method arranges each sequence into eight structured descriptors (global, four spatial parts, three temporal phases) and fuses their similarities under class-wise weights produced by a single LLM prompt. This dedicated design preserves topology, mitigates viewpoint drift, and consistently delivers the best accuracy on every NTU-RGBD split without back-propagation or synthetic augmentations.

In contrast, existing baselines exhibit limitations when applied to skeleton data. Gradient-based prompt tuning (TPT, DiffTPT) requires iterative optimisation that easily over-fits the small and highly imbalanced target streams. Prototype/attention schemes (AdaNPC, CALIP) compress each clip into a single holistic vector, discarding fine-grained joint dynamics and thus confusing actions that differ only in local motion. TDA partially addresses this with a dual cache, yet still operates at a single global scale and ignores temporal phase diversity. None of these methods explicitly model skeleton topology or spatial-temporal structure, which explains their inferior performance compared with the skeleton-aware retrieval strategy proposed here.

Table 7: Comparison with other TTA methods on NTU-RGBD datasets.

Method	NTU-RGBD 60		NTU-RGBD 120	
	55/5	48/12	110/10	96/24
TPT [24]	40.11	27.98	22.10	17.10
DiffTPT [6]	42.27	28.34	23.43	20.01
AdaNPC[31]	81.77	42.52	72.50	53.16
CALIP [7]	80.02	45.10	72.27	51.84
TDA[11]	82.84	43.95	74.92	54.60
Ours	85.46	45.22	77.60	56.83

C.3 Cache Update with Adapted Logits

To explore the impact of cache update strategies, we conducted an experiment where Skeleton-Cache is updated using the final adapted logits (post-fusion, as defined in Equation 12) instead of the zero-shot logits from the *PURLS* model. This approach leverages the enhanced predictions after combining cache-based similarity logits with the original model logits, potentially incorporating more refined confidence estimates. Table 8 compares the Top-1 accuracy of this adapted-logits strategy

Table 8: Top-1 accuracy (%) of Skeleton-Cache with cache updates using adapted logits versus zero-shot logits on NTU RGB+D datasets.

Split	SC (Zero-Shot Logits)	SC (Adapted Logits)	Difference
NTU60 55/5	85.46	85.77	+0.31
NTU60 48/12	45.22	44.66	-0.56
NTU120 110/10	77.60	77.84	+0.24
NTU120 96/24	56.83	56.90	+0.07

against the standard Skeleton-Cache (SC) approach, which uses zero-shot logits for cache updates, across four zero-shot splits of NTU RGB+D 60 and 120. The results show minimal differences, with adapted-logits accuracies of 85.77%, 44.96%, 77.84%, and 56.90% compared to 85.46%, 45.22%, 77.60%, and 56.83% for the standard approach. Notably, a slight performance decline is observed on the NTU60 48/12 split (44.96% vs. 45.22%). This may stem from reduced overall entropy in the adapted logits, as the fusion process (Section 3.2) enhances prediction confidence, potentially allowing

suboptimal samples to enter the cache despite the high-confidence selection criterion. These samples may not represent the class as effectively, impacting retrieval accuracy. The limited performance variation across other splits suggests that the high-confidence filtering mechanism (Section 3.2) is robust, mitigating significant deviations when using adapted logits.

D Analytical Visualizations

This section presents visualizations to illustrate the impact of Skeleton-Cache on prediction quality, focusing on confusion matrices, entropy shifts, and LLM-derived weight distributions. These analyses provide insights into how the method improves class discrimination and confidence in predictions.

D.1 LLM Weight Heatmap Analysis

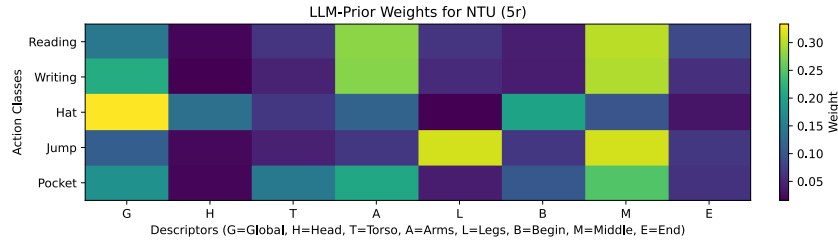


Figure 3: Heat-map visualisation of GPT-4-derived weights $w^{(c)}$. Columns correspond to the eight descriptors; rows correspond to unseen classes of NTU 55/5 split.

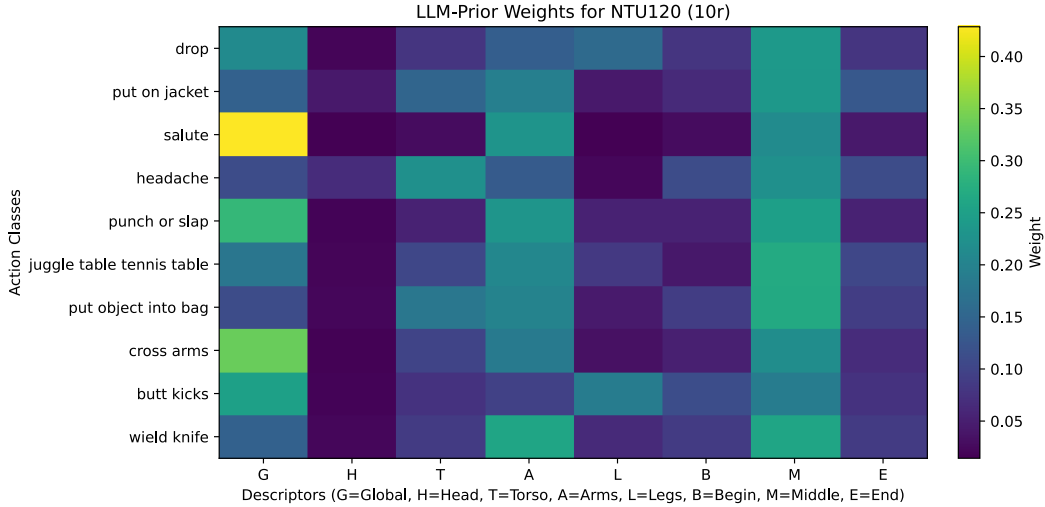


Figure 4: Heat-map visualisation of GPT-4-derived weights $w^{(c)}$. Columns correspond to the eight descriptors; rows correspond to unseen classes of NTU 110/10 split.

Figures 3 and 4 display heatmaps of the GPT-4-derived weights $w^{(c)}$ for the NTU 55/5 and 110/10 splits, respectively. These weights align closely with human intuition about actions. For example, in the NTU 55/5 split, *reading* and *writing* emphasize the *arms* descriptor, while *put on hat/cap* assigns significant weight (approximately 40%) to the *head*. Similarly, *jump up* focuses on *legs* and the *middle* temporal segment, reflecting the action’s peak motion phase. This alignment with commonsense semantics enhances inter-class discriminability by assigning distinct weight patterns to different actions, enabling the model to differentiate between similar motion patterns effectively.

625 D.2 Confusion Matrix Analysis

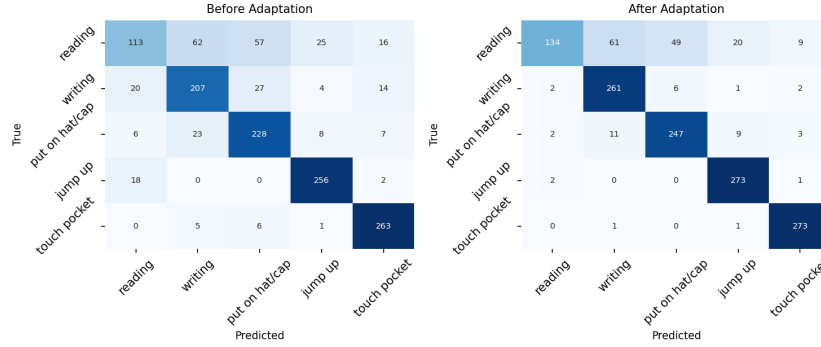


Figure 5: Comparison of confusion matrices on NTU 55/5 split. Adaptation with Skeleton-Cache sharpens the diagonal and reduces off-diagonal confusion, indicating improved class discrimination.

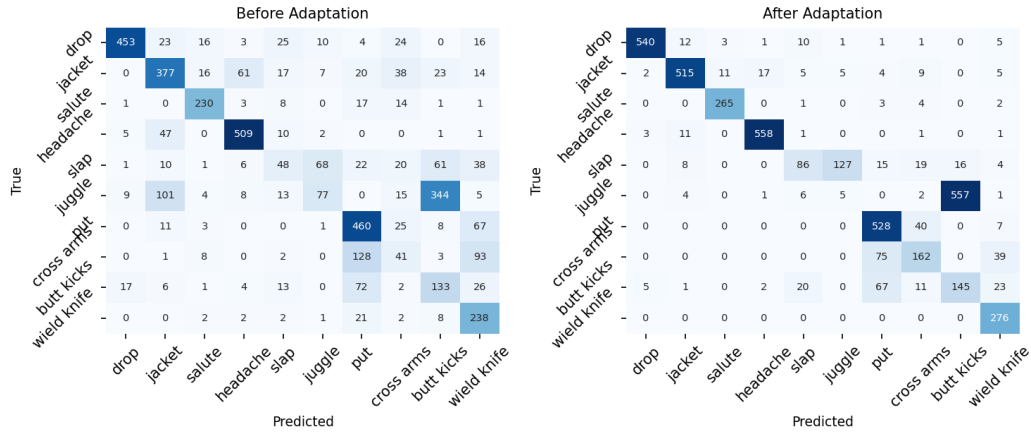


Figure 6: Comparison of confusion matrices on NTU 110/10 split. Adaptation with Skeleton-Cache sharpens the diagonal and reduces off-diagonal confusion, indicating improved class discrimination.

626 Figures 5 and 6 compare confusion matrices before and after adaptation with Skeleton-Cache for
 627 the NTU 55/5 and 110/10 splits. Before adaptation, the model struggles to distinguish between
 628 similar actions, such as *reading* and *writing*, due to their overlapping upper-body poses, resulting in
 629 significant off-diagonal confusion. Post-adaptation, the diagonal sharpens, and off-diagonal elements
 630 lighten, indicating that Skeleton-Cache enables differentiation of previously indistinguishable classes.
 631 However, this improvement introduces a trade-off: the model occasionally exhibits overconfidence,
 632 leading to misclassifications in some cases, as evidenced by residual off-diagonal elements.

633 D.3 Top-5 Prediction Changes

634 Figure 7 visualizes the confidence score shifts in the top-5 predictions for 10 unseen classes in the
 635 NTU 110/10 split. Skeleton-Cache significantly enhances prediction confidence, particularly for
 636 challenging zero-shot tasks. For instance, actions like *salute* and *cross arms* show marked increases
 637 in confidence for the correct class, reducing uncertainty and refining the prediction distribution.
 638 This boost in confidence underscores Skeleton-Cache’s ability to leverage structured descriptors and
 639 cache-based retrieval to improve recognition accuracy for unseen actions.

D.4 Per-Class Accuracy Changes Before and After Adaptation

To elucidate the impact of Skeleton-Cache (SC) on per-class performance, we visualize the Top-1 accuracy changes for the NTU RGB+D 55/5 and 110/10 splits before and after test-time adaptation with PURLS+SC (Section 3.2). Figure 8 compares the base PURLS accuracies against the adapted accuracies, highlighting the contribution of the cache-based retrieval and LLM-guided fusion.

For the NTU 55/5 split (Figure 8a), Skeleton-Cache consistently improves accuracy across all classes, with notable gains in *writing* (76.10% to 95.96%) and *reading* (41.39% to 49.08%). These improvements stem from the structured descriptors capturing fine-grained motion patterns (e.g., hand movements) and the LLM-guided weighting emphasizing relevant body parts (Section 3.2). For the NTU 110/10 split (Figure 8b), most classes show substantial gains, such as *cross arms* (14.86% to 58.70%) and *slap* (17.45% to 31.27%), benefiting from the cache’s ability to retrieve discriminative local patterns. However, *juggle* exhibits a significant drop (13.37% to 0.87%), likely due to its complex, multi-limb coordination, which challenges the cache’s retrieval when high-confidence exemplars are scarce. Overall, Skeleton-Cache enhances generalization for most classes, with occasional limitations for actions requiring intricate motion patterns.

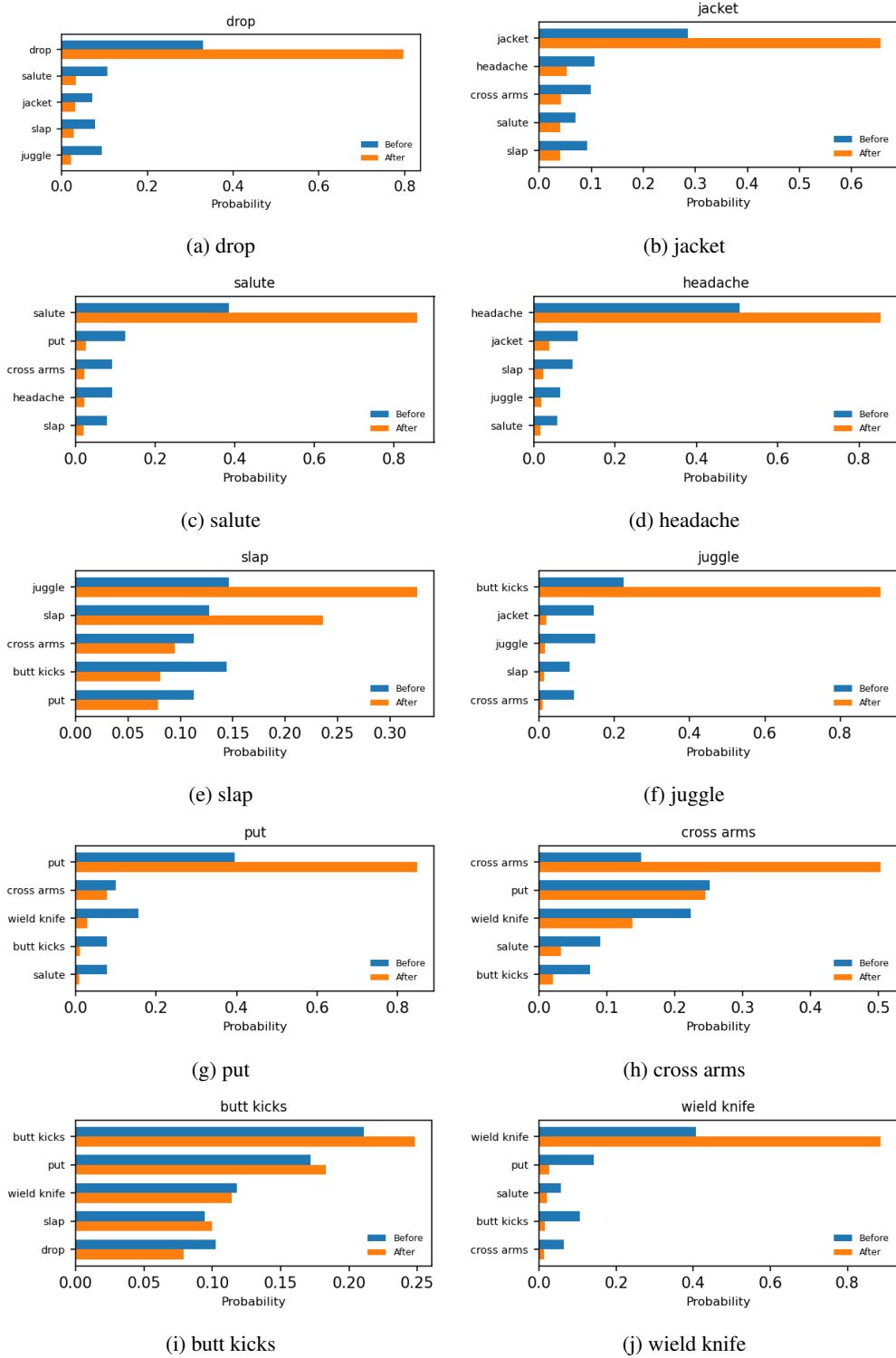
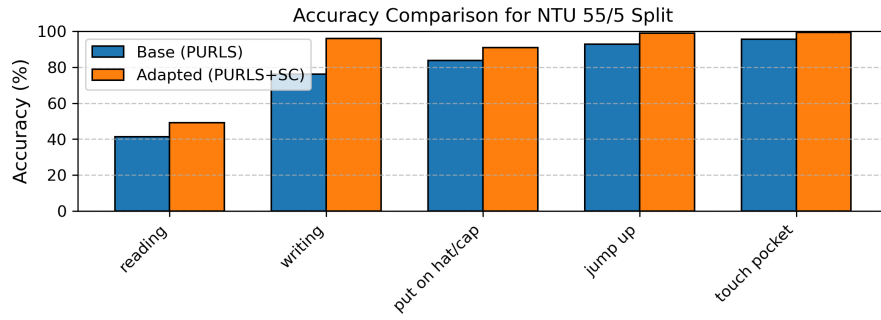
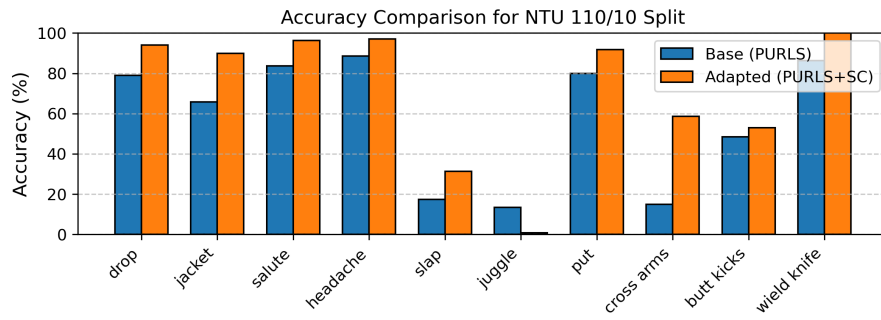


Figure 7: Visualization of Top-5 prediction changes before and after applying the Skeleton-Cache for 10 different unseen action classes. Each image displays the confidence score distribution shift when using our caching mechanism, demonstrating how the prediction accuracy improves for challenging zero-shot action recognition tasks.



(a) NTU 55/5 Split



(b) NTU 110/10 Split

Figure 8: Per-class accuracy comparison between base PURLS and adapted PURLS+SC for NTU 55/5 and 110/10 splits. Blue bars represent base accuracies, and orange bars represent adapted accuracies.