

PRISM: Learning a Shared Primitive Space for Transferable Skeleton Action Representation - Supplementary Material

Di Yang¹ Yaohui Wang^{2*} Shuai Shao¹ François Brémond³ Jiangtao Wang^{4*}

¹Suzhou Institute for Advanced Research, University of Science and Technology of China, China

²Shanghai Artificial Intelligence Laboratory, China

³Inria Center at Université Côte d’Azur, France

⁴School of Computing, Engineering and Digital Technologies, Teesside University, United Kingdom

Appendix

In this Appendix, we provide further details to enrich the experimental analysis presented in the main paper. Section A elaborates on more implementation details of our framework. Section B provides additional quantitative comparisons, extended analyses, qualitative results, to demonstrate the effectiveness of PRISM.

A. Experiment and Implementation Details

In this section, we firstly provide the details of the datasets used in this work. Secondly, we describe the architectural details of the key modules in PRISM.

A.1. Dataset Details

We conduct extensive experimental analysis on multiple datasets, including different tasks. We describe all the dataset details in the subsection and the Tab. 1 summarizes the datasets used in this work.

Posetics [15] is created on top of Kinetics-400 [3] and includes 142,000 real-world clips covering 320 action classes, each with 2D/3D skeletons. As the Posetics dataset manually merges the activities with the same motion but related to different objects (*e.g.*, the “eat piza”, “eat sushi” are merged into “eat”) to be more proper for human-centric activity recognition, the class distribution is naturally unbalanced in a long-tail distribution. We use it for pre-training and evaluation and report Top-1 and Top-5 accuracy.

Toyota Smarthome Trimmed (TST) [6] contains 31 fine-grained action classes over 16,115 videos, with strong long-tail and viewpoint / subject variations. We use officially provided 2D skeletons and report Mean Per-class Accuracy under Cross-Subject (CS), Cross-View1 (CV1), and Cross-View2 (CV2) protocols.

Toyota Smarthome Untrimmed (TSU) [5] extends the TST dataset focusing on frame-wise detection in long untrimmed sequences with dense multi-label annotations. We use officially provided 2D skeletons data and report per-frame mAP under Cross-Subject (CS) and Cross-View (CV) protocols.

Charades [13] contains unconstrained, overlapping activities in real-world indoor scenes. We extract 2D skeletons from RGB using existing pose estimation tool [14] and evaluate with per-frame mAP.

Mixamo [9] provides 3D motion sequences applied to stylized characters. We use it for training and evaluating the generation components (*i.e.*, Motion Decomposition Encoder and Decoder) of PRISM via cross-view/character motion transfer via synthetic ground truth.

NTU-RGB+D 120 (NTU-120) [11] is a widely-used skeleton-based action classification dataset. It contains 114,480 skeleton sequences recorded by RGBD camera in a laboratory setting. We evaluate PRISM on NTU-120 following Cross-Subject (CS) and Cross-Set (CSet). Although it is a well-established benchmark, its constrained setting is less aligned with our target real-world, multi-label, and long-tail tasks, we included the results in the Appendix for completeness.

A.2. Model Details

In this section, we describe the architectural details of the key modules in PRISM, including the Motion Decomposition Encoder and the Context-aware Encoder with Temporal Feature Fusion (TFF). The Tab. 2 summarizes the model architecture details.

Motion Decomposition Encoder and Decoder: The Motion Decomposition Encoder and the skeleton sequence Decoder, are built as the same autoencoder in [1, 16]. They are designed by multiple 1D temporal convolutions to process the skeleton sequences. To decode the skele-

*Corresponding authors

Dataset	Task Type	Used For	Training Stage	Evaluation Metric
Mixamo [9]	Generation (motion decomposition, cross-view transfer)	Primitive learning	Pre-train (Stage 1)	transfer quality
Posetics [15]	Action classification	Representation evaluation; transfer from Mixamo	Stage 2 (first transfer)	Top-1 / Top-5 accuracy
TST [6]	Action classification	Fine-grained recognition; transfer from Posetics	Stage 2 (first transfer)	Mean Per-class Accuracy (CS / CV1 / CV2)
TSU [5]	Multi-label frame-wise action detection	Action Detection; transfer from TST	Stage 3 (second transfer)	Per-frame mAP (CS / CV)
Charades [13]	Multi-label frame-wise action detection	Action Detection; transfer from TST	Stage 3 (second transfer)	Per-frame mAP
NTU-RGB+D 120 [11]	Action classification	General classification benchmark; transfer from Posetics	Stage 2 (first transfer)	Accuracy (CS / CSet)

Table 1. **Summary of datasets, tasks, and training stages used in PRISM.** PRISM is trained in a stage-wise manner: (1) primitive learning from motion generation on synthetic 3D data; (2) transfer to classification; (3) further transfer to frame-wise detection. No joint or multi-task training is used.

Module	Input	Output	Params	Used In	Description
Primitive Decomposition Encoder	$\mathbf{X} \in \mathbb{R}^{T \times J \times D}$	$\mathbf{A} \in \mathbb{R}^{T \times K}$	1.2M	Stage 1–3	TCN/MLP stack producing primitive coefficients
Primitive Basis $\{\mathbf{p}_k\}_{k=1}^K$	–	K learnable bases	0.05M	Stage 1–3	Shared motion atoms for decomposition and reconstruction
Primitive Decoder Decoder $_{\phi}$	\mathbf{A}	$\hat{\mathbf{X}}$	0.6M	Stage 1	Used only for Mixamo generation training
Context Encoder g_{ϕ}	\mathbf{X}	temporal features	3.4M	Stage 2–3	Lightweight TCN capturing local context
Temporal Feature Fusion (TFF) C_{ϕ}	$[\mathbf{A}, g_{\phi}(\mathbf{X})]$	\mathbf{H}	0.09M	Stage 2–3	Fuses primitive and contextual cues
Classification Head	$\text{Pool}(\mathbf{H})$	action label $\hat{\mathbf{y}}$	–	Stage 2	Global pooling + MLP
Detection Head	\mathbf{H}	frame-wise labels $\hat{\mathbf{Y}}$	–	Stage 3	Temporal MLP/TCN for multi-label detection

Table 2. **Summary of PRISM architecture components used across stages.** The model is trained in a stage-wise manner. Only the primitive space and decomposition encoder are shared across tasks.

Stage	Encoder	Decoder
Input	$\mathbf{X} \in \mathbb{R}^{T \times 2V}$	$\mathbf{Z} \in \mathbb{R}^{T' \times K}$
1	Conv(8, 64)	Upsample(2) + Conv(7, 160)
2	Conv(8, 96)	Upsample(2) + Conv(7, 64)
3	Conv(8, 160)	Upsample(2) + Conv(7, 2V)
Output	$\mathbf{A} \in \mathbb{R}^{T' \times K}$	$\hat{\mathbf{X}} \in \mathbb{R}^{T \times 2V}$

Table 3. Network details of the Decomposition Encoder.

ton sequence, the Decoder includes upsampling processes along the temporal dimension to reconstruct the skeleton sequences (see Tab. 3). Without specifically stated, we use $K = 160$ as the output feature size *i.e.*, the number of primitives. For real-world setting, our objective is to extract view-invariant motion primitives from 2D skeletons that lack depth information, which is a key challenge in our target scenarios. During pre-training, 3D Mixamo motions are rotated and projected to generate multiple 2D observations. The decomposition encoder operates entirely on 2D skeleton sequences, while the underlying 3D data is used only to synthesize multi-view supervision.

Context-aware Encoder and TFF: The context-aware encoder processes the input sequence to obtain context feature maps $\mathbf{M} \in \mathbb{R}^{T \times D}$. Specifically, for real-world evaluation on Posetics, TST, TSU, and Charades, we adopt SoTA backbone UNIK [15] as the context encoder with the same hyper-parameter settings [15]. To demonstrate the application of our framework in NTU-RGB+D 120 using 3D data in Tab. 6 in the main paper, we use current 3D SoTA Transformer backbone [7] as the context-aware encoder. To fuse motion primitives \mathbf{A} and contextual features \mathbf{M} , we employ

Temporal Feature Fusion (TFF), which is temporal modeling for the combined \mathbf{A} and \mathbf{M} . We adopt Dilated TCNs [4] as the architecture of TFF.

A.3. Training Details

In this section, we outline the training hyper-parameters and optimization settings.

Generation Block: We train the Decomposition Encoder and Decoder using reconstruction loss, physical constraints, sparsity, and static-motion disentanglement objectives for 300 epochs using Adam optimizer with learning rate $1e-3$ on the Mixamo dataset. The detailed physical-inspired constraint can be summarized as follows.

\mathcal{L}_{bone} enforces temporal consistency of bone lengths:

$$\mathcal{L}_{bone} = \sum_t \sum_{(i,j)} \|\mathbf{p}_i^t - \mathbf{p}_j^t\| - \ell_{ij}, \quad (1)$$

where i, j denote linked joints, while \mathcal{L}_{joint} penalizes violations of joint angle limits:

$$\mathcal{L}_{joint} = \sum_t \sum_i \max(0, \theta_i^t - \theta_i^{max}), \quad (2)$$

and \mathcal{L}_{acc} regularizes temporal smoothness to suppress non-physical acceleration:

$$\mathcal{L}_{acc} = \sum_t \|\mathbf{p}^{t+1} - 2\mathbf{p}^t + \mathbf{p}^{t-1}\|_2^2. \quad (3)$$

The λ_{rec} , λ_{orth} , λ_{sparse} , and λ_{phys} are set to 1.0, 0.1, 0.01, and 0.1 respectively unless stated otherwise.

Action Classification: For action classification, we use an initial learning rate of 0.1 for 65, 50 epochs with step

LR decay with a factor of 0.1 at epochs {45, 55}, {30, 40} for Posetics and TST respectively. Weight decay is set to 1×10^{-4} for training the Context-aware Encoder and TFF. 2D skeleton inputs are pre-processed with normalization and centering following [12]. For training with 3D data, we follow the same setting as [7]. We perform joint-bone two-stream fusion for all action classification evaluations, following [7].

Action Detection: For action detection tasks, we use an initial learning rate of 0.01 for 50 epochs with step LR decay with a factor of 0.1 at epochs {30, 40} for all the TSU and Charades datasets. Weight decay is set to 1×10^{-4} for final models. We adopt a temporal sliding window with sizes 64 frames along the untrimmed sequences for training the Context-aware Encoder and TFF. For such multi-label detection, we use BCE (binary cross entropy) loss, which computes classification loss separately for all classes in the same frame.

A.4. Computational Resources

All experiments are conducted on a single NVIDIA V100 GPU. The Motion Decomposition stage on the Mixamo dataset takes approximately 10 hours to train with a batch size of 64. Downstream perception tasks (classification and detection) require around 30–40 hours depending on dataset scale.

A forward pass through the Decomposition Encoder and the perception heads together requires 0.28 GFLOPs for a typical input sequence (20-30 frames), which is significantly lower than common vision backbones [2, 8]. During inference, PRISM processes over 1.5 skeleton frames per millisecond on a single V100 GPU, enabling efficient deployment for real-time or streaming scenarios.

Training is performed in a stage-wise manner: (1) primitive learning via motion generation on Mixamo, (2) transfer to classification tasks (Posetics, TST, NTU-120), and (3) a second transfer to frame-wise detection (TSU, Charades). No joint multi-task training is used.

B. Additional Results and Discussions

This section presents further evaluations of PRISM on additional datasets, ablation results on design choices, and qualitative results for action generation and perception.

B.1. Primitive Size and Sparsity

We investigate the influence of the primitive size and coefficient sparsity on downstream perception tasks. Specifically, we vary the number of primitives on the motion-related (K_m) and static (K_s) components. As shown in Table 5, setting of 160 primitives with $K_m = 128$ and $K_s = 32$ achieves the best performance on TST under the

Training Strategy	TST CV2 (%)	TSU CV (%)
End-to-End	69.6	25.8
Stage-wise (Ours)	72.5	28.1

Table 4. Comparison of training strategies under cross-view protocols. Stage-wise training leads to better structured representation and generalization.

Category	Configuration	TST CV2 (%)	TSU CV (%)
Primitive Size	$K_m=96, K_s=64$	70.2	26.5
	$K_m=160, K_s=0$	68.1	24.3
	$K_m=128, K_s=32$	72.5	28.1
	$K_m=144, K_s=16$	71.3	27.4
Sparsity Weight	$\lambda_{\text{sparse}} = 0.0$	69.8	26.1
	$\lambda_{\text{sparse}} = 0.001$	71.3	27.8
	$\lambda_{\text{sparse}} = 0.01$	72.5	28.1
	$\lambda_{\text{sparse}} = 0.1$	70.8	27.3

Table 5. Ablation study on primitive coefficient size allocation and sparsity control. Optimal performance is achieved by a balanced primitive design and moderate sparsity.

cross-view protocol, suggesting that this configuration offers a good trade-off between dynamic representation capacity and static disentanglement. Reducing K_m leads to insufficient motion expressiveness, while increasing it introduces redundancy.

We also evaluate in Tab. 5 the impact of enforcing coefficient sparsity. Using ℓ_1 regularization on **A** slightly improves performance by promoting disentanglement and interpretability, but overly strong sparsity harms accuracy due to loss of expressiveness. These results validate that a moderately sparse, structured coefficient space contributes to better generalization in real-world action recognition tasks.

B.2. Training Strategy Analysis

We investigate the effect of our stage-wise training strategy by comparing it against an end-to-end approach where three modules (*i.e.*, generation, classification and detection) are optimized jointly from scratch. In our default strategy, we first train the Motion Decomposition Encoder and Decoder using only reconstruction, sparsity, and physics losses. Once the primitive space is well formed, we freeze or partially fine-tune the encoder during different downstream perception tasks. This decoupled training ensures stable primitive learning and avoids interference from noisy task-specific gradients.

By contrast, the end-to-end strategy jointly optimizes the encoders, decoder, and perception heads from the beginning. While this setting reduces training stages, we observe unstable convergence and lower interpretability in the learned primitives, resulting in degraded performance on both classification and detection tasks. As shown in Tab. 4, the stage-wise strategy consistently achieves better accu-

Model	Backbone only	+ PRISM (Ours)
HD-GNC [10]	47.1	52.2
BlockGCN [17]	47.6	52.1
Hyper-GCN [18]	48.3	52.6
HD-GCN [10]	85.7	87.6
BlockGCN [17]	86.9	88.0
Hyper-GCN [18]	86.9	88.3
HD-GCN [10]	-	34.1
BlockGCN [17]	-	34.4
Hyper-GCN [18]	-	37.3

Table 6. Results on Posetics Top1 (top), NTU120 CS (mid), TSU CS (bot) with different backbone Context Encoders.

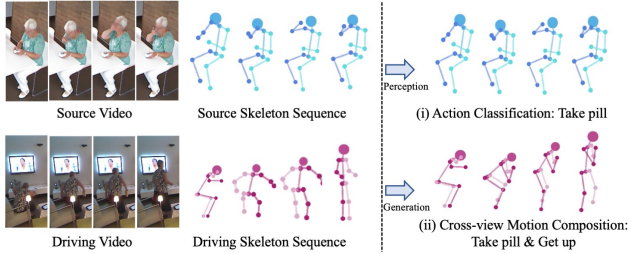


Figure 1. **Qualitative Results of PRISM.** Given two input skeleton sequences extracted from real-world custom source and driving videos (left), PRISM can generate and recognize the action (right) for the source sequence.

racy. This confirms the importance of first constructing a clean and semantically meaningful motion basis before using it for downstream understanding.

B.3. Application with More Context Backbones

To further clarify the role of PRISM, and reliance on specific encoders, we evaluate PRISM with multiple context backbones. PRISM is designed as a motion representation module rather than a backbone architecture. It can be integrated with different skeleton encoders by augmenting their contextual features with motion primitives learned from the decomposition model. Tab. 6 shows that PRISM consistently improves several recent backbones across datasets, indicating that the gains are not tied to a specific encoder design. The improvements remain consistent even when strong context encoders are used. This suggests that PRISM does not simply increase encoder capacity, but provides complementary motion structure that improves representation robustness.

B.4. More Qualitative Results

We include additional visualizations for reconstructed sequences, action classification, and cross-view motion composition outputs using real-world video in TST (see Fig. 1). Moreover, we show both action generation and perception

results with real-world videos (see the video attached in the supplementary material). These support the robustness of PRISM in diverse tasks.

B.5. Remarks on 2D Kinematic Constraints

The kinematic (physics-inspired) constraints in our decomposition model are applied on projected 2D skeleton sequences. Since perspective projection may introduce mild geometric distortions (e.g., foreshortening), quantities such as bone length are not strictly preserved across views. In practice, these terms mainly act as regularizers that encourage temporally stable and consistent motion patterns in the projected domain. Empirically, we observe that this formulation is sufficient for learning robust motion primitives under multi-view projections. Exploring depth-aware constraints or 3D-consistent formulations could be an interesting direction for future work.

References

- [1] Kfir Aberman, Rundi Wu, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Learning character-agnostic motion for motion retargeting in 2d. *ACM TOG*, 2019. 1
- [2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *ICCV*, 2021. 3
- [3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1
- [4] Rui Dai, Srijan Das, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca, and Francois Bremond. Pdan: Pyramid dilated attention network for action detection. In *WACV*, 2021. 2
- [5] Rui Dai, Srijan Das, Saurav Sharma, Luca Minciullo, Lorenzo Garattoni, Francois Bremond, and Gianpiero Francesca. Toyota smarhome untrimmed: Real-world untrimmed videos for activity detection. *IEEE TPAMI*, 2022. 1, 2
- [6] Srijan Das, Rui Dai, Michal Koperski, Luca Minciullo, Lorenzo Garattoni, Francois Bremond, and Gianpiero Francesca. Toyota smarhome: Real-world activities of daily living. In *ICCV*, 2019. 1, 2
- [7] Jeonghyeok Do and Munchurl Kim. Skateformer: skeletal-temporal transformer for human action recognition. In *ECCV*, 2024. 2, 3
- [8] K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3D residual networks for action recognition. In *ICCV*, 2017. 3
- [9] Adobe Systems Inc. Mixamo. <https://www.mixamo.com>. <https://www.mixamo.com>. Accessed: 2018-12-27., 2018. 1, 2
- [10] Jungho Lee, Minhyeok Lee, Dogyoon Lee, and Sangyoun Lee. Hierarchically decomposed graph convolutional networks for skeleton-based action recognition. In *ICCV*, 2023. 4

- [11] J. Liu, A. Shahroudy, M. Perez, G. Wang, L. Y. Duan, and A. C. Kot. Ntu rgb+d 120: A large-scale benchmark for 3D human activity understanding. *IEEE TPAMI*, 2020. [1](#), [2](#)
- [12] Dario Pavlo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In *CVPR*, 2019. [3](#)
- [13] Gunnar A. Sigurdsson, Gül Varol, X. Wang, Ali Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. [1](#), [2](#)
- [14] Di Yang, Rui Dai, Yaohui Wang, Rupayan Mallick, Luca Minciullo, Gianpiero Francesca, and Francois Bremond. Selective spatio-temporal aggregation based pose refinement system: Towards understanding human activities in real-world videos. In *WACV*, 2021. [1](#)
- [15] Di Yang, Yaohui Wang, Antitza Dantcheva, Lorenzo Garattoni, Gianpiero Francesca, and Francois Bremond. Unik: A unified framework for real-world skeleton-based action recognition. In *BMVC*, 2021. [1](#), [2](#)
- [16] Di Yang, Yaohui Wang, Antitza Dantcheva, Quan Kong, Lorenzo Garattoni, Gianpiero Francesca, and Francois Bremond. Lac - latent action composition for skeleton-based action segmentation. In *ICCV*, 2023. [1](#)
- [17] Yuxuan Zhou, Xudong Yan, Zhi-Qi Cheng, Yan Yan, Qi Dai, and Xian-Sheng Hua. Blockgcn: Redefine topology awareness for skeleton-based action recognition. In *CVPR*, 2024. [4](#)
- [18] Youwei Zhou, Tianyang Xu, Cong Wu, Xiaojun Wu, and Josef Kittler. Adaptive hyper-graph convolution network for skeleton-based human action recognition with virtual connections. In *ICCV*, 2025. [4](#)