STARS: Self-supervised Tuning for 3D Action Recognition in Skeleton Sequences

Supplementary Material

A. 3-Stage Design

An alternative pretraining method is to follow MAE-CT [4] and tune the encoder in 3 stages. Figure 1 shows the 3-stage design. Initially, when we initialize the projector and predictor modules, we freeze the encoder weights. In the second stage, we exclusively train the projector and predictor modules until they can effectively differentiate between different sequences using the NNCLR approach. Finally, in the third stage, we fine-tune the encoder weights using layer-wise learning rate decay. One motivation for this staged approach is the idea that the NNCLR head's random weights could interfere with the representation quality by mapping the features into a random space, disrupting the learned structure. However, our findings challenge this assumption. The t-SNE visualization in Fig. 2 demonstrates that even with random NNCLR head weights, the cluster structure in the encoder's output space remains intact in the new representation space. Furthermore, we observed with STARS-3stage that replicating this three-stage process not only increases training time but also leads to a drop in final accuracy. As a result, we use a two-stage design in our proposed STARS method.

B. Semi-supervised Evaluation Results

In semi-supervised evaluation protocol, we follow previous works [5, 7, 8] and fine-tune the pretrained encoder in addition to a post-attached classifier while given a small fraction of the training dataset. Specifically, the performance on the NTU-60 is reported while using 1% and 10% of the training set. Since the training portions are selected randomly, we report the result averaged over 5 different runs as the final result. As shown in Tab. 1, STARS is more effective in all scenarios. Specifically, while using 1% of the training data, leading to an increase in accuracy for the MAMP baseline by 3.1% and 4.2% in cross-subject and cross-view evaluations, respectively.

Method	NTU-60			
	XSub		XView	
	(1%)	(10%)	(1%)	(10%)
Other pretext tasks:				
LongT GAN [16]	35.2	62.0	-	-
ASSL [10]	-	64.3	-	69.8
Contrastive Learning:				
MS^{2}L [6]	33.1	65.1	-	-
ISC [11]	35.7	65.9	38.1	72.5
3s-CrosSCLR [5]	51.1	74.4	50.0	77.8
3s-Colorization [13]	48.3	71.7	52.5	78.9
CMD [7]	50.6	75.4	53.0	80.2
3s-Hi-TRS [2]	49.3	77.7	51.5	81.1
3s-AimCLR [3]	54.8	78.2	54.3	81.6
3s-CMD [7]	55.6	79.0	55.5	82.4
CPM [14]	56.7	73.0	57.5	77.1
Masked Prediction:				
SkeletonMAE [12]	54.4	80.6	54.6	83.5
MAMP [8]	66.0	88.0	68.7	91.5
Masked Prediction + Contrastive Learning:				
PCM^3 [15]	53.1	82.8	53.8	77.1
STARS-3stage (Ours)	68.6	**88.2**	72.5	**91.8**
STARS (Ours)	**69.1**	88.0	**72.9**	**91.8**

Table 1. Performance comparison on the NTU-60 dataset under the semi-supervised evaluation protocol, with the final performance reported as the average of five runs.

C. Ablation Hyper-parameters

Tab. 2 shows the hyperparameters used in DINOTuning strategy. For simplicity and because of limitation in resources, we used only two global views and did not use any local views in DINO. As shown in Tab. 3, we can see that incorporating local views led to a small improvement in performance. However, it came at the cost of significantly more resources. To be specific, we introduced two local views that randomly trimmed a section of the sequence between 40% and 80% and fed it only to the student network. With these additional views, we had to reduce the batch size to 16 and double the training time. Consequently, in our other experiments, we stuck to using only global views. We also used Sinkhorn-Knopp centering [1] a KoLeo regularizer [9] to help the convergence. Tab 6 shows the hyperparameters used in MoCoTuning. Similar to previous approaches [3, 5], we used 32K as the queue size, 0.999 for the momentum and 0.07 for the temperature.

Method	10-NN	20-NN	40-NN
w/o local views	77.4	77.1	77.0
w/ local views	77.8	78.0	77.6

Table 3. Effect of including local views on DINOTuning.

C.1. Algorithm psuedo code

Algo. 1 demonstrates the process in PyTorch-style pseudo code.

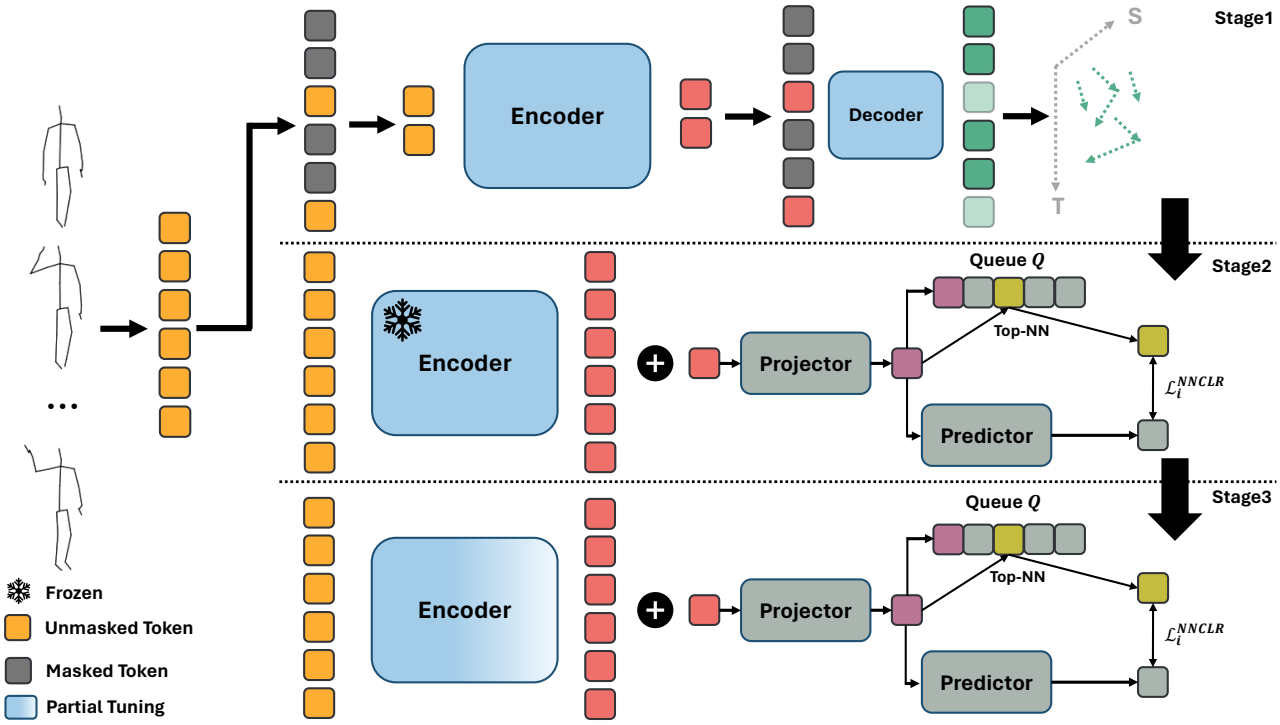

Figure 1. The overall pipeline of our proposed STARS-3stage framework. The first stage uses MAMP [8] to reconstruct the motion of masked tokens. The second stage keeps the encoder parameters frozen and trains parameters of the projector and predictor using a contrastive learning approach. After these parameters have converged to well-separated clusters, the third stage involves partial-tuning of the encoder parameters.

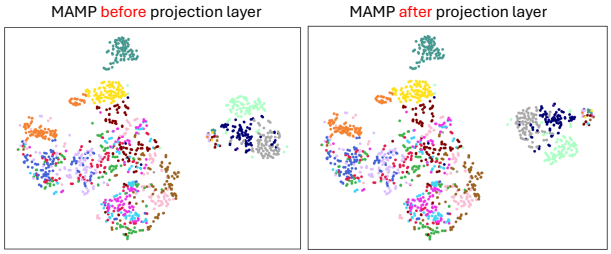

Figure 2. Comparison between MAMP's output vectors before and after using a projection layer with random weights.

Hyperparameter	Value
Learning rate	0.001
Batch size	32
Augmentations	Mirroring & Rotation
Centering	Sinkhorn Knopp
KoLeo weight	0.1
# Global views	2
# Local views	0
Student temperature	0.1
Teacher temperature	0.04
Teacher momentum	0.996

Table 2. DINOTuning hyperparameters for ablation study in tuning strategy design.

D. Additional Ablation Studies

Combining MAMP and NNCLR.: One idea to tune the encoder in second stage is to combine NNCLR with MAMP and use both. Tab 4 compares this tuning stage with the proposed STARS. By including MAMP in the second stage of tuning, although it improves the final representation of encoder compared to the baseline (MAMP), it cannot perform as effective as STARS.

Training NNCLR from scratch: Table 5 presents the K-NN evaluation results for training the transformer from scratch using the NNCLR method for 300 epochs. Without augmentation, the model struggles to select positive samples in contrastive learning that truly represent the same actions. This happens because the encoder starts with random weights, which lack meaningful cluster separation. Consequently, the encoder fails to fully use the advantages of NNCLR in the second stage of STARS, resulting in poorer performance.

Algorithm 1 PyTorch-style pseudo-code of contrastive tuning in the second stage.

```
# f: MAMP Encoder. Only second-half is tuned.
# g: Projector. Batch normalization module
# h: Predictor. 2 layer MLP, hidden size 4096, output 256
# Q: Queue with length of 32,768

for x in loader:
   y = f(x) # encoder forward pass
   z = g(y) # projection forward pass
   p = h(z) # prediction forward pass
   z, p = normalize(z, dim=1), normalize(p, dim=1)
   nn = top_nn(z, Q) # finding nearest-neighbor sample in Q
   loss = L(nn, p)
   loss.backward()
   optimizer.step()
   update_queue(Q, z)


def top_nn(z, Q):
   similarities = z @ Q.T
   idx = similarities.max(dim=1)
   return Q[idx]

def L(nn, p, temperature=0.07):
   logits = nn @ p.T
   logits /= temperature # sharpening
   labels = torch.arange(p.shape[0])
   loss = cross_entropy(logits, labels)
   return loss
```

Method	NTU-60	
	XSub	XView
MAMP (Baseline)	63.1	80.3
STARS	**79.9**	**88.6**
MAMP + NNCLR	74.6	86.5

Table 4. Ablation study on second stage of tuning. The performance is evaluated on the NTU-60 XSub and NTU-60 XView datasets under the KNN evaluation protocol (K=1).

K	NTU-60	
	XSub	XView
1	37.6	30.5
2	35.8	28.7
5	39.9	32.3
10	41.2	33.6

Table 5. Training the transformer encoder using NNCLR method from scratch. The performance is evaluated on the NTU-60 XSub and NTU-60 XView datasets under the KNN evaluation protocol.

Using naive MAE instead of MAMP: Tab. 7 and Tab.8 present a comparison of K-NN and few-shot evaluations for a variant that uses naive MAE instead of MAMP in the first stage. While tuning in the second stage leads to significant improvements, the overall performance remains lower because MAE performs worse than MAMP in the initial stage.

Effect of Batch Size. Table 9 reports the effect of varying the effective batch size on K-NN classification accuracy.

Hyperparameter	Value
Learning rate	0.001
Batch size	32
Augmentations	Mirroring & Rotation
Queue size	32,768
Momentum	0.999
Temperature	0.07

Table 6. MoCo hyperparameters for ablation study in tuning strategy design.

Method	NTU-60	
	XSub	XView
MAMP	63.1	80.3
STARS	79.9	88.6
MAE	44.1	43.7
STARS-MAE	53.5	55.2

Table 7. Ablation study on K-NN evaluation by changing the first-stage of STARS to naive MAE.

Method	1-shot	2-shot	5-shot
MAMP	47.6	44.4	48.4
STARS	63.5	62.2	65.7
MAE	35.0	31.8	34.2
STARS-MAE	41.5	37.9	40.6

Table 8. Ablation study on first few-shot evaluation by changing the first-stage of STARS to naive MAE.

Batch size	K-NN Acc. (%)
16	76.7
32	78.9
64	78.3
128	79.9

Table 9. Impact of effective batch size (i.e., number of negative samples) on K-NN accuracy, evaluated on NTU60-XSub. (K=1)

Increasing the batch size, which corresponds to sampling more negatives, generally improves accuracy, with the best performance observed at a batch size of 128. This highlights the importance of large negative sets for robust representation.

Setting	K-NN Acc. (%)
$\alpha = 1.0$	74.3
$\alpha = 0.2$	83.7

Table 10. Effect of tuning strength α on K-NN accuracy with $K = 10$, evaluated on NTU60-XSub.

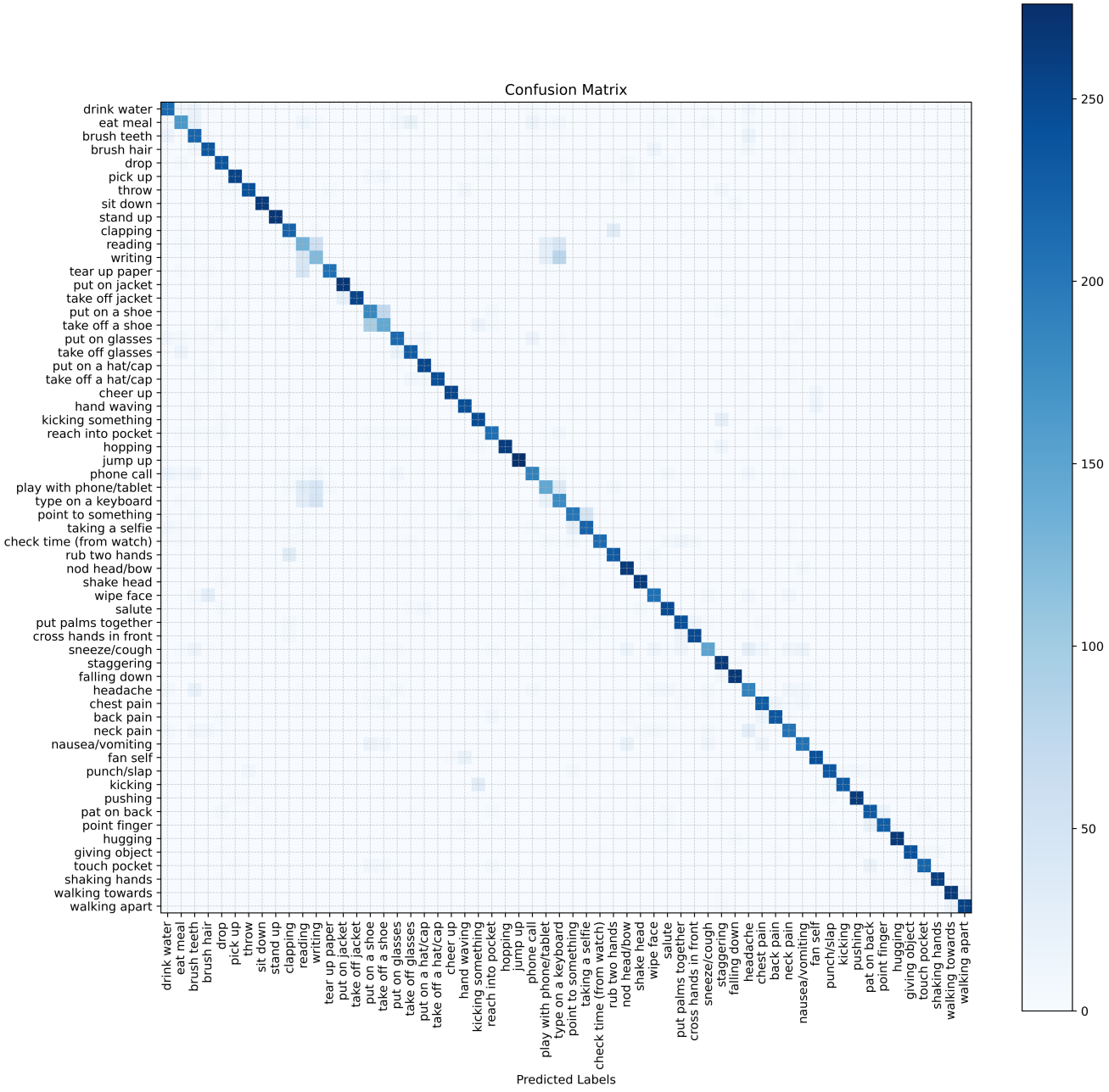

Figure 3. Confusion matrix of KNN evaluation in NTU-60 XSub dataset (k=10).

Effect of layer decay. As shown in Fig. 4 of the main paper, applying layer decay with $\alpha = 0.2$ yields the best performance. For comparison, we also evaluate the case of $\alpha = 1$, where all layers are tuned uniformly. Table 10 compares the effect of different tuning strengths on K-NN accuracy with $K = 10$. While full tuning ($\alpha = 1.0$) achieves moderate performance, reducing the tuning strength to $\alpha = 0.2$ substantially improves accuracy, suggesting that partial tuning helps preserve more generalizable representations.

Method	Parameters (M)	MACs (G)
MAMP	10.34	250.4
STARS	8.43	606.1

Table 11. Comparison of model size and computational cost between MAMP and STARS.

Parameters and MACs. Tab. 11 shows that STARS requires fewer parameters than MAMP, as it employs an MLP-based decoder instead of a transformer decoder. However, its MACs are significantly higher because training involves two forward passes for the contrastive loss as well as the additional computation needed to identify nearest neighbors and apply the loss.

E. Confusion Matrix

Fig. 3 illustrates the confusion matrix under KNN evaluation protocol when K=10 on NTU-60 XSub dataset. The errors depicted in the figure can be classified into two distinct categories. Firstly, there are errors stemming from a lack of contextual information. For instance, when only a skeletal sequence is provided, actions like "play with phone/tablet" might be misinterpreted as "reading" and "writing." Secondly, there are errors arising from subtle movements, such as distinguishing between "clapping" and "hand rubbing," which pose challenges for the model in differentiation. In summary, these errors manifest due to either insufficient context or the intricacy of distinguishing minute actions, highlighting the complexities inherent in the task.

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[2] Yuxiao Chen, Long Zhao, Jianbo Yuan, Yu Tian, Zhaoyang Xia, Shijie Geng, Ligong Han, and Dimitris N Metaxas. Hierarchically self-supervised transformer for human skeleton representation learning. In *European Conference on Computer Vision*, pages 185–202. Springer, 2022.

[3] Tianyu Guo, Hong Liu, Zhan Chen, Mengyuan Liu, Tao Wang, and Runwei Ding. Contrastive learning from extremely augmented skeleton sequences for self-supervised action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 762–770, 2022.

[4] Johannes Lehner, Benedikt Alkin, Andreas Fürst, Elisabeth Rumetshofer, Lukas Miklautz, and Sepp Hochreiter. Contrastive tuning: A little help to make masked autoencoders forget. *arXiv preprint arXiv:2304.10520*, 2023.

[5] Linguo Li, Minsi Wang, Bingbing Ni, Hang Wang, Jiancheng Yang, and Wenjun Zhang. 3d human action representation learning via cross-view consistency pursuit. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4741–4750, 2021.

[6] Lilang Lin, Sijie Song, Wenhan Yang, and Jiaying Liu. MS^{2}L: Multi-task self-supervised learning for skeleton based action recognition. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2490–2498, 2020.

[7] Yunyao Mao, Wengang Zhou, Zhenbo Lu, Jiajun Deng, and Houqiang Li. CMD: Self-supervised 3d action representation learning with cross-modal mutual distillation. In *European Conference on Computer Vision*, pages 734–752. Springer, 2022.

[8] Yunyao Mao, Jiajun Deng, Wengang Zhou, Yao Fang, Wanli Ouyang, and Houqiang Li. Masked motion predictors are strong 3d action representation learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[9] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Spreading vectors for similarity search. *arXiv preprint arXiv:1806.03198*, 2018.

[10] Chenyang Si, Xuecheng Nie, Wei Wang, Liang Wang, Tieniu Tan, and Jiashi Feng. Adversarial self-supervised learning for semi-supervised 3d action recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 35–51. Springer, 2020.

[11] Fida Mohammad Thoker, Hazel Doughty, and Cees GM Snoek. Skeleton-contrastive 3d action representation learning. In *Proceedings of the 29th ACM international conference on multimedia*, pages 1655–1663, 2021.

[12] Wenhan Wu, Yilei Hua, Ce Zheng, Shiqian Wu, Chen Chen, and Aidong Lu. SkeletonMAE: Spatial-temporal masked autoencoders for self-supervised skeleton action recognition. In *2023 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 224–229. IEEE, 2023.

[13] Siyuan Yang, Jun Liu, Shijian Lu, Meng Hwa Er, and Alex C Kot. Skeleton cloud colorization for unsupervised 3d action representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13423–13433, 2021.

[14] Haoyuan Zhang, Yonghong Hou, Wenjing Zhang, and Wanqing Li. Contrastive positive mining for unsupervised 3d action representation learning. In *European Conference on Computer Vision*, pages 36–51. Springer, 2022.

[15] Jiahang Zhang, Lilang Lin, and Jiaying Liu. Prompted contrast with masked motion modeling: Towards versatile 3d action representation learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 7175–7183, 2023.

[16] Nenggan Zheng, Jun Wen, Risheng Liu, Liangqu Long, Jianhua Dai, and Zhefeng Gong. Unsupervised representation learning with long-term dynamics for skeleton based action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.