Hierarchically Self-Supervised Transformer for Human Skeleton Representation Learning (Supplementary Material)

Yuxiao Chen[1], Long Zhao[2], Jianbo Yuan[3], Yu Tian[3], Zhaoyang Xia[1], Shijie Geng[1], Ligong Han[1], and Dimitris N. Metaxas[1]

[1] Rutgers University, [2] Google Research, [3] ByteDance Inc.

1 Implementation Details of Pre-training

The implementation details for pre-training are introduced in this section.

Data Augmentation. During training, data augmentations are applied to both the spatial and temporal domains of a skeleton sequence. Specifically, the spatial augmentation method includes: (1) shearing [3]: the skeleton of each frame is sheared at a random angle by applying a linear transformation matrix whose elements are uniformly sampled from the range [-0.5, 0.5]; (2) adding noise: a noise sampled from the uniform distribution with the range [-0.2, 0.2] is added to 5 randomly sampled joints at each frame. The temporal augmentation method contains: (1) time interpolation [6]: it generates a new skeleton sequence by linearly interpolating the positions of each joint between consecutive time with a random scale whose range is from 0 to 1; (2) temporal cropping: it temporally crops a random portion of frames from the original sequence as the training sample. Note that for each training sample, only one spatial and one temporal augmentation method are used.

Model Pre-training. We pre-train the model using the Nvidia A100 GPU. The batch size is set to 256. The dropout rate [7] is set to 0.2, and the L2 weight decay is set to 0.01. The Adam optimizer [2] with the learning rate of 4×10^{-4} is used for training. We use linear learning rate warmup for the first 1,000 iterations followed by polynomial decay. The model is trained for 30,000 iterations. The window size, stride, and dilate of the sliding window are set to 7, 7, and 2, respectively. For the training samples which contain two subjects, we divide them into two samples, each of which contains the skeleton sequence of one subject.

2 Implementation Details of Downstream Tasks

This section describes the implementation details for each downstream task.

Action Recognition. For the action recognition task, we use the entire Hi-TRS model as the encoder. The outputs from the V-TRS model are fed to a linear classifier (a fully-connected layer) to predict action categories. To train

our network, the cross-entropy loss and the Adam Optimizer with a learning rate of 4×10^{-4} are used. The batch size is set to 128. The window size, stride, and dilate of the sliding window are set as 7, 7, and 2, respectively. If the encoder is pre-trained, the learning rate is divided by 2 at the 20th, 25th, and 30th epochs. The training process is terminated at the 50th epoch. If the encoder is randomly initialized, the learning rate is divided by 2 at the 40th, 50th, and 60th epochs. The training process is terminated at the 70th epoch.

Action Detection. In the action detection task, each skeleton sequence contains multiple different actions [4]. It is formulated as a per-frame classification problem following the setup in [4]. Specifically, if a frame belongs to an action of the c-th category, its label is c; otherwise, its label is 0. For a skeleton sequence with n frames, a sliding window with size 7, stride 1, and dilate 4 is employed to temporally crop n clips from the sequence. The n-th obtained clip contains about one-second surrounding information of the n-th frame. It is fed to the F-TRS and C-TRS models to extract its clip-level embedding. Then, a linear classifier (a fully-connected layer) is used for predicting the action category of the n-th frame. The models are trained to minimize the cross-entropy loss by using the Adam Optimizer. The learning rate and batch size are set as 4×10^{-4} and 256, respectively. The model is trained for 5 epochs.

Motion Prediction. The model used in this task consists of an encoder and a decoder. The encoder is our proposed Hi-TRS. For the decoder, we use the LSTM-based architecture proposed in [5], and we implement it based on its Pytorch implementation [1]. We train the model to minimize the mean squared error loss. We set the batch size as 128 and set the dropout rate of the decoder as 0.5. We train the model for 25 epochs by using the Adam Optimizer. The initial learning rate is set to 1×10^{-4} and is decayed by 0.5 at the 20th epoch.

3 Implementation Details of Previous Methods

Previous work [8,3] only evaluated their methods on the action recognition task. The reported results for the action detection and motion prediction tasks are based on our implementation. In this section, we introduce the implementation details of these methods.

MCC [8]. Since the official implementation of this method is not publicly available, we implement it by ourselves according to the details in [8]. Specifically, the ST-GCN encoder [9] and momentum memory bank [1] used by MCC are implemented based on their officially released codes [2] [3]. During pre-training, we completely follow the hyper-parameters settings of MCC [8], such as the batch size, memory bank size, and learning rate. When conducting downstream tasks, the pre-trained encoders are fine-tuned following the same settings as our method for a fair comparison.

[1] `https://github.com/enriccorona/human-motion-prediction-pytorch`
[2] `https://github.com/yysijie/st-gcn`
[3] `https://github.com/facebookresearch/moco`

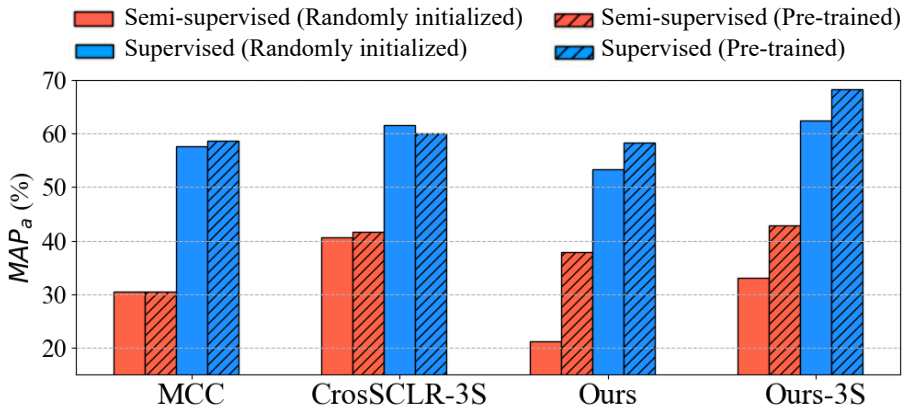

Fig. 1. mAP_a (%) results on the action detection task.

Table 1. Results (%) for the action detection task on the PKUMMD Part I subset under supervised and semi-supervised settings.

Method	100% data		10% data	
	mAP_a	mAP_v	mAP_a	mAP_v
MCC	58.6	62.3	30.5	44.5
CrosSCLR-3S	60.1	72.4	41.6	60.9
Ours	58.4	66.3	37.9	54.8
Ours-3s	**68.2**	**76.0**	**42.9**	**62.9**

CrosSCLR [3]. We pre-train the encoders by using its official implementation[4], and fine-tune the pre-trained encoders following the same settings as ours for action detection and motion prediction.

4 Additional Results of Action Detection

The results for action detection when using the mAP_a metric are shown in Figure 1. We can observe the similar findings when the mAP_v metric is used (shown in the main paper). Additionally, the exact numbers are reported in Table 1.

Qualitative analysis. We show two action detection examples in Figure 2. We can see that the temporal boundaries of the actions detected by our methods are more accurate than the baselines (MCC and CrosSCLR-3S). The results further demonstrate that the prior knowledge learned through our different-level pretext

[4] `https://github.com/LinguoLi/CrosSCLR`

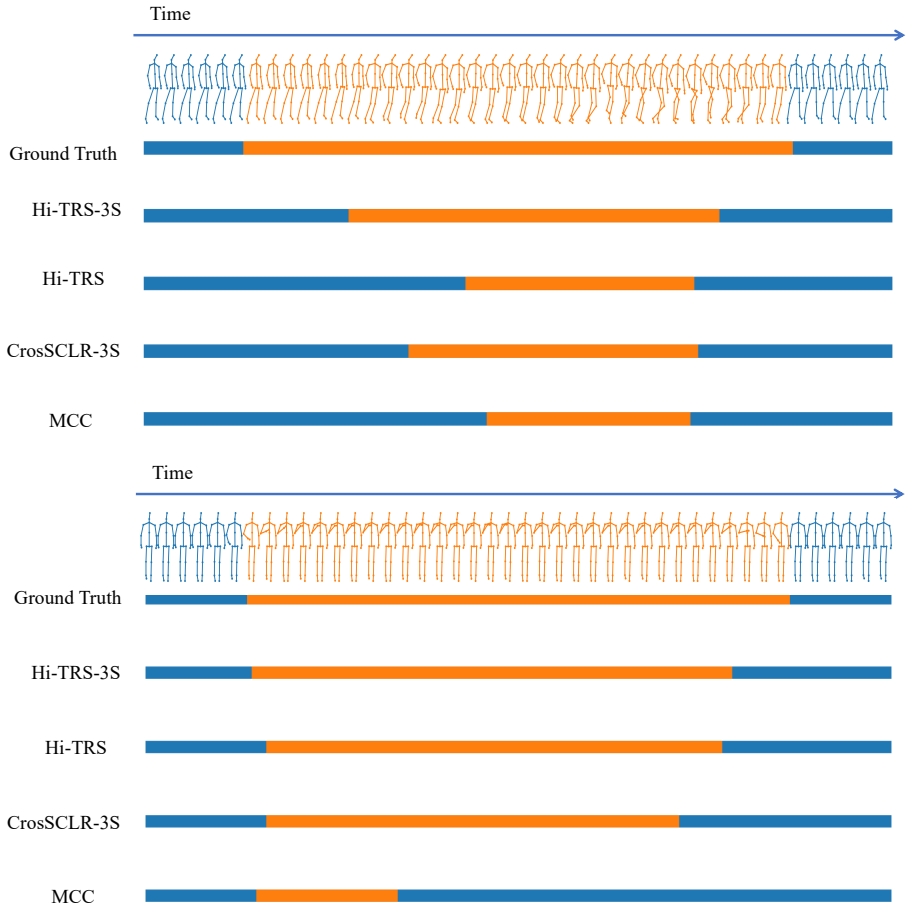

Fig. 2. Examples for the action detection task on the PKUMMD Part I subset. The blue regions represent that there are no actions. The orange regions denotes that the correspondent actions are detected. **Top:** An example showing the action detection results of the one foot jumping action. **Bottom:** An example showing the action detection results of the touching chest action.

tasks is more useful for extracting short-term information than that learned by using the baselines' video-level pretext tasks.

5 Additional Results of Motion Prediction

The correspondent numbers of the main paper's Figure 3 Right are reported in Table 2.

Qualitative analysis. We visualize two cases of motion prediction in Figure 3. We can observe that the MCC and CrosSCLR models fail to predict the

Table 2. Results for the motion prediction task on the NTU-60 cross-subject benchmark under the supervised and semi-supervised settings.

Method	100% data	10% data
	MPJPE ↓	MPJPE ↓
MCC	90.4	109.8
CrosSCLR	89.3	109.3
Ours	**85.6**	**104.7**

Table 3. Results (%) of the ablation study for action detection

Pre-trained Level	-	F	C	V	F+C	F+V	C+V	F+C+V
mAP_a	53.4	55.5	55.3	55.9	55.6	57.9	56.3	**58.4**
mAP_v	63.2	64.1	63.9	63.9	65.1	64.0	64.0	**66.3**

fine-grained movement patterns, such as the motion of the legs and right arm in Figure 3 **Top** and **Bottom**, respectively. This may because they are only pre-trained to learn coarse video-level motion patterns [8] or discriminative information [3]. By contrast, our proposed hierarchical pre-training framework help our encoders capture the motion patterns at different level. As a result, our method can produce more accurate motion prediction.

6 Additional Results of Ablation Study

We report the results of all model variances for action detection in Table 3. We observe that pre-training at lower level is beneficial to higher level downstream tasks. It is consistent with the ablation study for other downstream tasks (see Table 4 of the paper). Additionally, the results show that leveraging higher level pre-training also improves lower level downstream tasks.

References

1. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9729–9738 (2020)
2. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
3. Li, L., Wang, M., Ni, B., Wang, H., Yang, J., Zhang, W.: 3d human action representation learning via cross-view consistency pursuit. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4741–4750 (2021)
4. Liu, C., Hu, Y., Li, Y., Song, S., Liu, J.: Pku-mmd: A large scale benchmark for continuous multi-modal human action understanding. arXiv preprint arXiv:1703.07475 (2017)

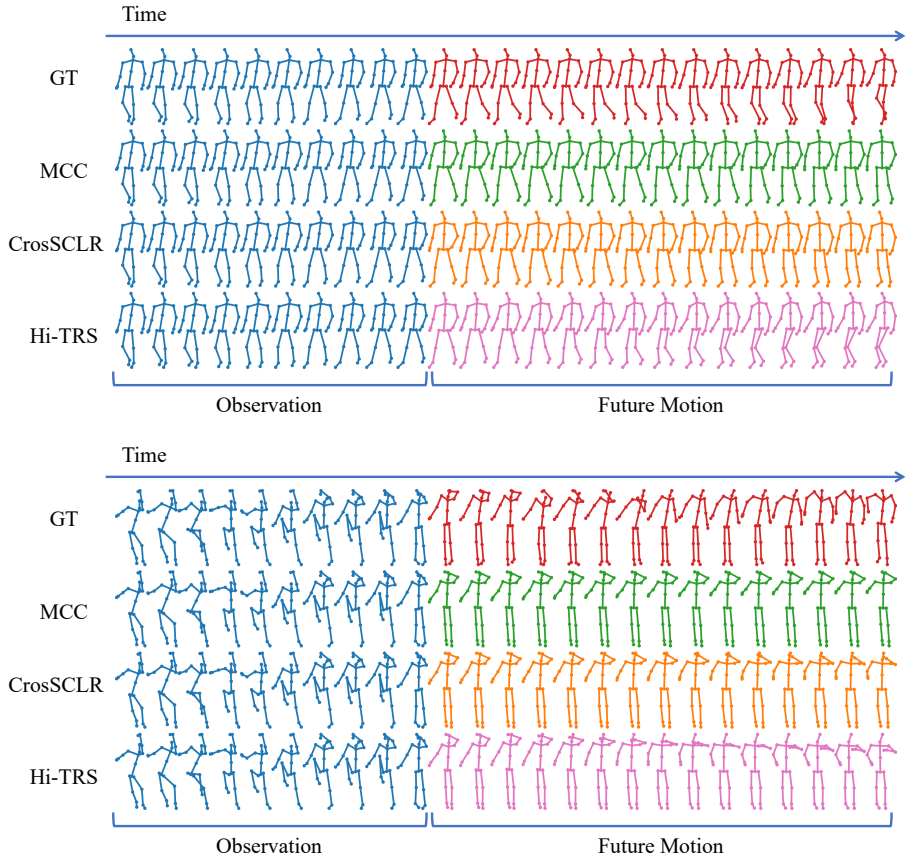

Fig. 3. Examples for the action prediction task on the NTU-60 cross-subject benchmark under the supervised setting. "GT" means ground truth.

5. Martinez, J., Black, M.J., Romero, J.: On human motion prediction using recurrent neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2891–2900 (2017)
6. Nunez, J.C., Cabido, R., Pantrigo, J.J., Montemayor, A.S., Velez, J.F.: Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. Pattern Recognition **76**, 80–94 (2018)
7. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
8. Su, Y., Lin, G., Wu, Q.: Self-supervised 3d skeleton action representation learning with motion consistency and continuity. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13328–13338 (2021)
9. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Thirty-second AAAI conference on artificial intelligence (2018)