STMT: A Spatial-Temporal Mesh Transformer for MoCap-Based Action Recognition

1. Model Architecture

The input to surface field convolution is a set of vertices in shape $N \times C$ and the coordinates of a set of centroids of size $N' \times 3$, where N is the number of vertices, C is the feature dimension, and N' is the number of centroids. The outputs are groups of vertex sets of size $N' \times K \times C$, where each group corresponds to a local region and K is the number of vertices in the nearest neighborhood of centroid vertices. To learn both intrinsic and extrinsic features, we sample the K nearest neighbors in Geodesic and Euclidean space, respectively. Then each local region is abstracted by its centroid and local feature that encodes the centroid's neighborhood. Output data size is $N' \times C'$. We use two surface field convolution blocks. In the first block, we sample 64 centroids with 8 nearest neighbors. In the second block, we sample 32 centroids with 8 nearest neighbors. The feature dimension C and C' equal 256. For the hierarchical spatial-temporal transformer, we use 2 offset-attention layers for the intra-frame attention module. The inter-frame attention module contains a self-attention block with 8 heads.

2. Datasets

2.1. Data Pre-Processing

As most of the existing skeleton-based and point-cloud-based baselines are for single-class classification, we only use the MoCap sequences with single-class annotations. There are 6,570 and 21,653 sequences for KIT and BABEL after data cleaning. Both datasets use the SMPL-H sequences from AMASS [8]. For 3D skeleton-based baselines, we use the pre-processed 3D skeletons provided by the official BABEL dataset [9]. It predicted the 25-joint skeleton used in NTU RGB+D [10] from the vertices of the SMPL-H mesh. The process involves human efforts to identify the vertices in the SMPL+H mesh that correspond to these joints in the NTU RGB+D skeleton. For the data pre-processing of noisy pose estimations on NTU-RGB+D dataset, we apply the state-of-the-art body pose estimation model VIBE [5] on videos of NTU RGB+D to obtain 3D mesh sequences. Skeleton and point cloud representations are derived from the estimated meshes to train the baseline modes. There are 45,035 mesh sequences after pre-processing. We follow the cross-view evaluation protocol (*i.e.*, use the samples of camera 1 for testing and samples of cameras 2 and 3 for training [10]). We manually convert the 72-dimensional pose parameters from the estimated SMPL sequences into the standardized NTU-RGB+D skeleton format. For point cloud-based models, we directly use the mesh vertices as model input. For our *STMT* model, the mesh vertices along with their adjacent matrices are used as input. As MoCap sequences have variant lengths, we sample 24 frames from each MoCap sequence. We use farthest point sampling to sample 128 vertices from each frame.

2.2. Dataset Licenses

AMASS [8]: https://amass.is.tue.mpg.de/license.html
BABEL [9]: https://babel.is.tue.mpg.de/license.html
NTU-RGB+D [10]: https://rose1.ntu.edu.sg/dataset/actionRecognition

3. Experiments

3.1. Training Details

For skeleton-based baselines, we use the official implementations of 2s-ACGN, CTR-GCN, and MS-G3D from [11], [1], and [7], respectively. We train models for 250 epochs with a batch size of 64. The other hyper-parameters are the same as the hyper-parameters used in NTU-RGB+D dataset. For point-cloud-based baselines, we use the official implementations of PSTNet, Sequential-PointNet, P4Transformer from [4], [6], and [3]. Our *STMT* model is pre-trained using Adam optimizer with a learning rate of 0.0001 for 120 epochs. The batch size is 128. We use equal weights ($\lambda_1 = \lambda_2 = 0.5$) for masked vertex reconstruction loss and future frame prediction loss. The pre-training stage takes 18 hours on 8 Tesla V100 (32GB) GPUs, and the fine-tuning stage takes 1.5 hours on 4 Tesla V100 (16GB) GPUs.

3.2. Computational Efficiency and Memory Usage

We evaluate the computational efficiency and memory usage, *i.e.*, the number of parameters and GFLOPs, of our

Method	# Frames	# Params (M)	GFLOPs	Running time per clip (ms)	Top-1 (%)
P4Transformer [2]	24	44.21	65.94	27.59	62.15
STMT (Lightweight)	6	**10.55**	**59.59**	**15.41**	**63.50**

Table 1. Comparison of Computational Efficiency and Memory Usage.

method. As mesh's local connectivity cannot be directly aggregated in the temporal domain, we cannot use temporal stride as in P4Transformer [2]. Therefore, we compare our ablated model which only takes 6 frames as input with P4Transformer, which is the best point-cloud-based baseline. We can see that our light-weighted model has much fewer parameters and smaller GFLOPs, and it can outperform P4Transformer by 1.35% in terms of Top-1 accuracy.

3.3. Transfer Learning Ability of the Pre-Trained Model

To evaluate the transfer ability of the pre-trained model, we train our model on KIT and test it on the joint dataset BABEL, which combines more than 15 datasets. We report the results with and without the proposed self-supervised pre-training method (*i.e.*, Maksed Vertex Modeling and Future Frame Prediction) in Table 2. We observe that our pre-training method can learn robust and generalized features, even when the model is not pre-trained on the target domain.

Method	Top-1 (%)	Top-5 (%)
Rand.init.	41.90	68.37
STMT Pre-Training	**43.16**	**69.04**

Table 2. Analysis of the Transfer Ability.

4. Limitations

Although we validated our method on BABEL, which combines more than 10 datasets, it still suffers from the long-tailed problem. Some action classes have very few sequences and may not represent the intra-class variances. Therefore, potential dataset biases need to be addressed before deploying the model.

References

[1] Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. https://github.com/Uason-Chen/CTR-GCN, 2021. 1

[2] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2021. 2

[3] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. https://github.com/hehefan/P4Transformer, 2021. 1

[4] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan S. Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. https://github.com/hehefan/Point-Spatio-Temporal-Convolution, 2021. 1

[5] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. VIBE: Video inference for human body pose and shape estimation. In *CVPR*, 2020. 1

[6] Xing Li, Qian Huang, Zhijian Wang, Zhenjie Hou, and Tianjin Yang. Sequentialpointnet: A strong frame-level parallel point cloud sequence network for 3d action recognition. https://github.com/XingLi1012/SequentialPointNet, 2021. 1

[7] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. https://github.com/kenziyuliu/MS-G3D, 2020. 1

[8] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, 2019. 1

[9] Abhinanda R. Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J. Black. BABEL: Bodies, action and behavior with english labels. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 722–731, June 2021. 1

[10] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A large scale dataset for 3D human activity analysis. In *CVPR*, 2016. 1

[11] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. https://github.com/abhinanda-punnakkal/BABEL/tree/main/action_recognition, 2019. 1