No Pain, Big Gain: Classify Dynamic Point Cloud Sequences with Static Models by Fitting Feature-level Space-time Surfaces (Supplementary Materials)

Jia-Xing Zhong, Kaichen Zhou, Qingyong Hu✉, Bing Wang, Niki Trigoni, Andrew Markham
Deparment of Computer Science, University of Oxford
{jiaxing.zhong, rui.zhou, qingyong.hu, bing.wang, niki.trigoni, andrew.markham}@cs.ox.ac.uk

1. NTU-RGBD: Effectiveness on Large-scale Data

NTU-RGBD has two subsets, *i.e.*, NTU-RGBD 60 [21] and NTU-RGBD 120 [9]. We evaluate our model on the former since it is included in more comparison results of previous researches. NTU-RGBD 60 is a large-scale dataset for 3D action recognition, with 56880 RGBD videos of 60 action categories. The data is converted into point cloud sequences with the implementation of PSTNet [4].

Comparison Following the official data partition [21], cross-subject and cross-view scenarios are individually adopted as two splits. As shown in Table 1, Kinet is superior to the others, with an accuracy of 92.3% in the cross-subject split. In the cross-view protocol, Kinet comes a close second (with 96.4%) after 96.5% of PSTNet.

Methods	Modalities	Accuracy (%)	
		Cross-subject	Cross-view
SkeleMotion [2]	Skeleton	69.6	80.1
GCA-LSTM [11]	Skeleton	74.4	82.8
Attention-LSTM [10]	Skeleton	77.1	85.1
AGC-LSTM [24]	Skeleton	89.2	95.0
AS-GCN [8]	Skeleton	86.8	94.2
VA-fusion [32]	Skeleton	89.4	95.0
2s-AGCN [23]	Skeleton	88.5	95.1
DGNN [22]	Skeleton	89.9	96.1
MS-G3D [13]	Skeleton	91.5	96.2
HON4D [19]	Depth Map	30.6	7.3
SNV [31]	Depth Map	31.8	13.6
HOG^2 [18]	Depth Map	32.2	22.3
Li *et al*. [7]	Depth Map	68.1	83.4
Wang *et al*. [26]	Depth Map	87.1	84.2
MVDI [29]	Depth Map	84.6	87.3
3DV-Appearance [28]	Point Cloud	80.1	85.1
3DV-Motion [28]	Voxel	84.5	95.4
3DV-Full [28]	Point Cloud + Voxel	88.8	96.3
P4Transfomers [3]	Point Cloud	90.2	96.4
PSTNet [4]	Point Cloud	90.5	**96.5**
Kinet	Point Cloud	**92.3**	96.4

Table 1. *Quantitative results achieved on* NTU-RGBD 60.

2. Implementation Details

For the sake of reproducibility, we elaborate the implementation as detailed as possible and the source code will be released soon.

2.1. Matrix Inversion

Equation (6) in the main body of this paper is to fit the group-wise ST-surface via the closed-form least-squared solution:

$$[A_k^{T*}, b_k^*] = (F_{i,k}^{(t)T} W_{i,k}^{(t)} F_{i,k}^{(t)})^{-1} F_{i,k}^{(t)T} W_{i,k}^{(t)} \boldsymbol{\tau}_{i,k}^{(t)}, \tag{1}$$

where the weight matrix $W_{i,k}^{(t)} = diag(w_{1,k}^{(\tau)}, ..., w_{|N_i^{(t)}|,k}^{(\tau)}) \in \mathbb{R}^{|N_i^{(t)}| \times |N_i^{(t)}|}$, the feature matrix $F_{i,k}^{(t)} = [(\mathbf{f}_{1,k}, 1), .., (\mathbf{f}_{|N_i^{(t)}|,k}, 1)] \in \mathbb{R}^{|N_i^{(t)}| \times (d+1)}$ and the time vector $\boldsymbol{\tau}_{i,k}^{(t)} \in \mathbb{R}^{|N_i^{(t)}|}$.

Note that we do not utilize the default implementation of matrix inversion on Tensorflow [1] since the default implementation using the LU decomposition is specially designed for large matrices. In our settings, matrices are relatively small in each group and we adopt home-brew approaches to matrix inversion. Given an $n \times n$ square matrix $M \in \mathbb{R}^{n \times n}$, we inverse the matrix according to its size.

In the case of extremely small matrices ($n \leq 4$), we directly inverse the matrix with element-wise calculations. For example, let $M = \left(\begin{smallmatrix} m_1 & m_2 \\ m_3 & m_4 \end{smallmatrix}\right) \in \mathbb{R}^{2\times 2}$. The inversion of this matrix is $M^{-1} = \frac{1}{m_1 m_4 - m_2 m_3}\left(\begin{smallmatrix} m_4 & -m_2 \\ -m_3 & m_1 \end{smallmatrix}\right)$, where $m_1 m_4 - m_2 m_3$ is clipped to the minimal value of 1×10^{-6} for the numerical stability of singular matrices. In terms of medium matrices ($4 < n \leq 16$), we recursively inverse the partitioned matrices based on the aforementioned element-wise operations for small matrices. For the case $n > 16$, we leverage Cholesky decomposition [6] to solve the matrix inversion.

2.2. Network Structures

2.2.1 Kinet: MLP-based Backbone (PointNet++)

The MLP-based methods separately model every point with shared multi-layer perceptions (MLPs), followed by a symmetric aggregation (*e.g.*, max pooling) to fuse order-invariant information. In this paper, we choose PointNet++ [20] as our MLP-based backbone for static point clouds. For a fair comparison, we keep the number of parameters in each layer identical to FlickerNet [15], one of the state-of-the-art gesture recognition networks on sequential point clouds.

2.2.2 Kinet: Graph-based Backbone (DGCNN)

Unlike the MLP-based backbones independently capturing point-level representations, graph-based methods model point-wise interactions by regarding a point cloud as a graph, of which each point is the vertex and edges is established upon the neighboring distribution of these points. For the graph-based paradigm, we conduct experiments on the static backbone of DGCNN [27] and keep the default settings of layer-wise parameters including the feature aggregation of the channel-wise additions in [27].

2.2.3 Kinet: Conv.-based Backbone (SpiderCNN)

Different from the other two paradigms using graphs or MLPs, the conv.-based models directly devise the convolution particularly for unstructured point clouds. We adopt SpiderCNN [30] as our convolution-based static backbone and keep the same layer-wise settings including the feature aggregation of **concatenation** and **top-k pooling**.

2.3. Training Configurations

The proposed framework is implemented with Tensorflow [1]. All experiments are conducted on the NVIDIA DGX-1 stations with Tesla V100 GPUs. During the training stage, the hyper-parameters are $batch\_size = 16$, $base\_learning\_rate = 0.001$, with an Adam optimizer [5]. The number of training epochs depends on various datasets: 200 epochs on *NvGesture*/*SHREC'17*, 150 epochs on *MSRAction-3D*, and 20 epochs on *NTU-RGBD*. For training stability, we first train the static backbone (spatial stream) until convergence and then freeze its weights to individually optimize the dynamic branch (temporal stream). To make full use of the features in various scales, multiple layers are connected to output the final results. As for the hyper-settings of Kinet itself, we set the ratio of feature reduction as 50%, group-wise dimensions $d = 4$, temporal radius $\Delta t = 1$, and spatial radius $\Delta r = 0.5$, respectively.

2.4. Point Activation Clouds

This concept stems from FlickerNet [15], which indicates the highlighted points of a model. Likewise, P4Transformer [3] and PSTNet [4] have the similar visualization concepts of the attentional values and the convolutional outputs, respectively. For readability, we formulate the concept of Point Activation Clouds (PACs) for the proposed two-stream framework.

Given a frame P_t of point clouds at the t^{th} time step, denote the activated feature vector of the i^{th} point $p_i^{(t)}$ in the l^{th} layer as $\boldsymbol{\phi}_l(p_i^{(t)})$. The PAC of a centroid $p_i^{(t)}$ in this layer is defined as:

$$PAC_l(p_i^{(t)}) = \max_{\phi_l \in \boldsymbol{\phi}_l} \max_{p_j^{(\tau)} \in N_i^{(t)}} \phi_l(p_j^{(\tau)}) \,, \tag{2}$$

where $N_i^{(t)}$ is the space-time neighbor set of $p_i^{(t)}$. Thus, the larger PACs reflect the greater activated values for these centroids, which highlights the discernible outputs in the l^{th} layer.

In the main body of this paper, we visualize the PACs of the last layer before the first max-pooling operation in the static stream of PointNet++ [20] and the corresponding layer in our dynamic branch.

2.5. Point-wise Scene Flow Estimation of Flow-based Baselines

In Section 4.3 of the main body, a self-supervised framework *Justgo* [17] is trained to estimate scene flow and we adopt it as an additional input in the setting 3). Note that Kinet does not require scene flow and setting 3) serves as the flow-based baseline.

2.5.1 Self-supervised Training Details

Based on the model pre-trained on *Flythings3D* [14], we train the scene flow estimator with the following hyper-parameters: $batch\_size = 8$, $radius = 5$, $flip\_prob = 0.5$, $base\_learning\_rate = 0.001$, with an Adam optimizer [5]. The balancing weight of the nearest neighbor loss and the cycle loss is set as 1:1.

2.5.2 Inference

The scene flow is first extracted by applying the trained *Justgo* model. Then, we scale the estimated scene flow to the same range of the raw point clouds, so that a static model with default hyper-parametric settings can be directly applied to the input modality of scene flow. Note that the number of parameters in a trained *Justgo* (excluding the parameters of the Adam optimizer) achieves 3.54M. For a 16-frame snippet with 2048 points per frame, the snippet-wise computational complexity of scene flow extraction is 154.29G FLOPs. The computational overhead of scene flow estimation is considerable.

2.5.3 Visualization

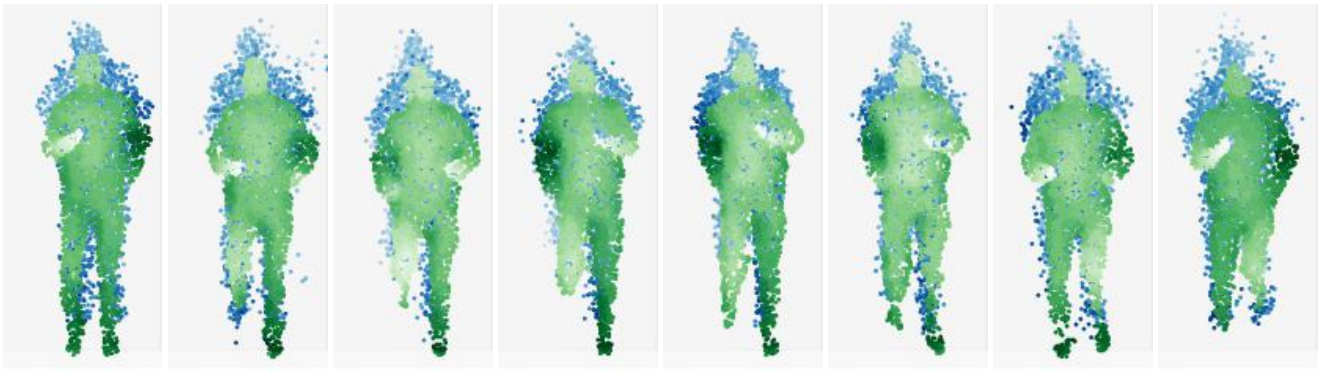
As depicted in Figure 1b, scene flow is decently estimated between two consecutive frames (as depicted in Figure 1a) with the self-supervised network. It is observed that key motions of the runner's legs are well captured but the interpolation results for the runner's head are insufficiently accurate: scene flow estimation without explicit supervision is a highly challenging task. As one of the input modalities for a vanilla two-stream model, the scaled scene flow in Figure 1c highlights the legs' movements, still preserving crucial dynamic information.

3. Detailed Experimental Results

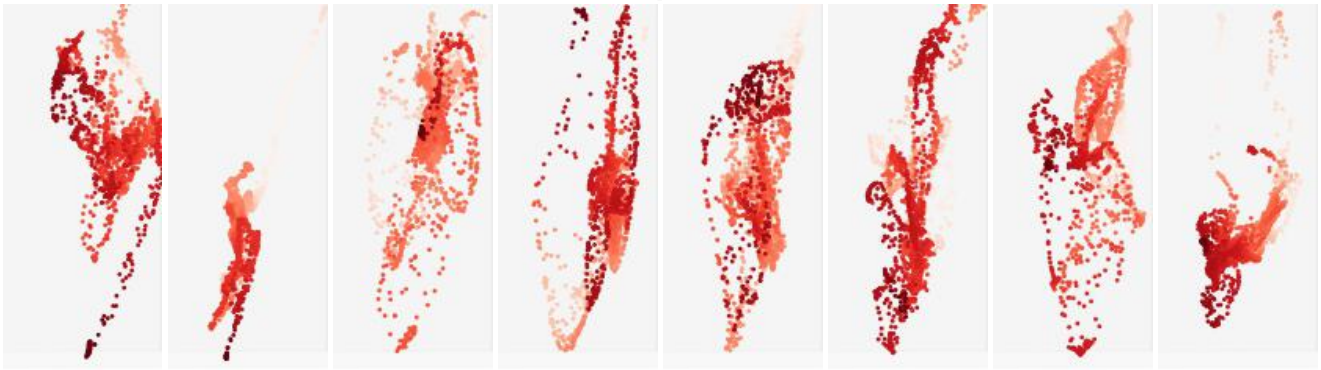
3.1. Quantitative Computational Costs

In section 4.3 of the main body, we conduct 3 groups of experiments: 1) Directly feed videos into the static model; 2) Fuse the static model and the dynamic branch; 3) Ensemble classification scores from two static models, one is trained on the raw point clouds, while the other is trained on estimated scene flow. As shown in Table 2, the extra input modality of estimated scene flow in setting 3) (+S, Flow-based Two Streams) considerably improves the accuracy of the three static models (Static PointNet++, DGCNN and SpiderCNN) to a level comparable to the state-of-the-art. However, the scene flow estimator and another flow-based classifier almost triple the number of parameters. Even worse, the estimation of scene flow introduces more than 150G FLOPS of extra calculations since the point-wise dense predictions are required between every two consecutive frames. For setting 2) (+Ours, Kinet), it is observed that our kinematic representations consistently increase

(a) Point Cloud



(b) Interpolation



(c) Scaled Flow

Figure 1. *Estimated point-wise scene flow of the flow-based baselines on* MSRAction-3D. The darker color indicates the greater depths (in point clouds) or the larger motions (in scene flow). (a) is the input of raw point clouds. For the frame interpolation in (b), the green points are the ground-truth point clouds, while the blue points are the interpolation results based on the last frame and the estimated scene flow. For the estimated scene flow in (c), it is normalized to the same scale as raw point clouds.

the accuracy of the static predictions by 5.99%∼9.04% relative gains. By utilizing the kinematic representations, the FLOPS only increases to 5.83G∼15.29G, and the number of parameters increases by 0.59M∼1.08M. These computing overheads are negligible and make the fused model extremely lightweight.

Model	FLOPS (G)	#PARAMS (M)	Accuracy (%)
Static PointNet++ [20]	4.82	2.13	84.30
Flow-based Two Streams (PointNet++)	163.93	7.79	90.23
Kinet (Pointnet++)	10.35	3.20	91.92
Static DGCNN [27]	5.33	5.25	84.18
Flow-based Two Streams (DGCNN)	164.95	14.04	89.56
Kinet (DGCNN)	5.83	5.85	89.82
Static SpiderCNN [30]	13.40	8.03	83.49
Flow-based Two Streams (SpiderCNN)	181.09	19.60	89.23
Kinet (SpiderCNN)	15.29	8.62	88.54
Meterornet [12]	1.70	17.60	88.21
PSTNet [4]	54.09	8.44	89.90
P4Ttransformer [3]	40.38	42.07	89.56

Table 2. *Quatative results of parameter number, FLOPS and accuracy on 16-frame* MSRAction-3D.

3.2. Per-class Performance Gains

Following Liu *et al*. [12], we report the performance gains over the original static model on *MSRAction-3D*. Under the same experimental configurations as the main body, we take 16 frames as an input unit and sample 2048 points for each frame. By comparing with the baseline (*PointNet++*) accuracy of setting 1), we report the performance changes of setting 2) and 3) in per-class action categorization.

As illustrated in Figure 4a, our dynamic branch significantly improves the classification accuracy of the static model. Noticeably, the categories of "High Throw" and "Hand Catch" show more than 30% absolute performance gains and most of the other performance changes achieve at least 20% improvements. There is only one category ("Horizontal Arm Wave") with about 6% performance decline. Figure 4b demonstrates that the extra input modality of estimated scene flow also boosts the overall performance. It has a large performance drop of more than 10% in the category of "High Arm Wave" and limited performance gains on many classes. Compared with the raw scene flow, our kinematic approach has high robustness and considerable improvements because it does not rely on the inaccurate estimation of scene flow.

3.3. Confusion Matrices

Following prior researches [15, 16] on point-based gesture recognition, we utilize confusion matrices to report the detailed performance on *NvGesture* and *SHREC'17*. A confusion matrix (a.k.a. an error matrix) shows whether a classifier is confounded by two categories, of which each column represents the instances in an actual class and each row represents the examples in a predicted category.

In the main body, we evaluate our model on *SHREC'17* under two input cases: 1) based on the entire video with backgrounds of the performer's body (w/o BBox); 2) based on the area of hand skeletons inside the bounding boxes (w/ BBox). It is observed in Figure 5 that the absence of bounding boxes leads to misclassification for more categories (*e.g*., "Tap-1" and "Swipe Down-1") but the total number of error instances is quite small. From this, it is observed that Kinet manifests high robustness to backgrounds.

As depicted in Figure 6, our framework is able to distinguish an overwhelming majority of gestures. However, it fails in some extremely challenging cases. For example, the classifier confuses the category "Show Two Fingers" with "Push Two Fingers Away" possibly because point clouds are scarce and sparse in the part of fingers, which makes it difficult to capture subtle differences between these two gestures.

4. More Visualizations

4.1. Point-level Sequential PACs

Due to the page limitation in the main body, we visualize the depth videos and dynamic PACs in Figure 6 with the format of animations. In case the PDF reader cannot display the animations normally, we provide the sequential images as shown in Figure 2.

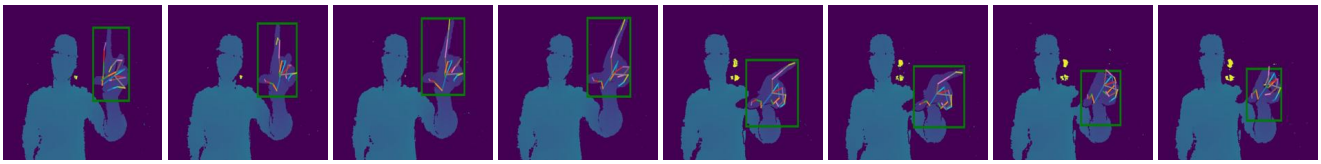
Unlike most of the prior works only focusing on the background-free cases, we adopt two input settings to verify whether Kinet can capture useful movements and ignore meaningless ones: 1) w/ BBox (Figure 2a) - used by a majority of existing

methods for high accuracy, based on the area inside the bounding boxes of hand skeletons without background interference; 2) w/o BBox (Figure 2d) - raw videos with noisy backgrounds (the performer's body). With bounding boxes removing noisy backgrounds, the two streams work complementary - the static branch (Figure 2b) highlights the main parts (the palms) of spatial appearances, whereas the temporal representations (Figure 2c) capture key motions, such as the movement of fingers and wrists. In the case with redundant backgrounds (without bounding boxes), the static stream (Figure 2e) excessively focuses on the large yet useless background portions (the performer's body), while the temporal stream (Figure 2f) captures the moving parts (arms and fingers). Inevitably, the temporal stream also highlights several redundant points of the performer's shaking head by mistake.

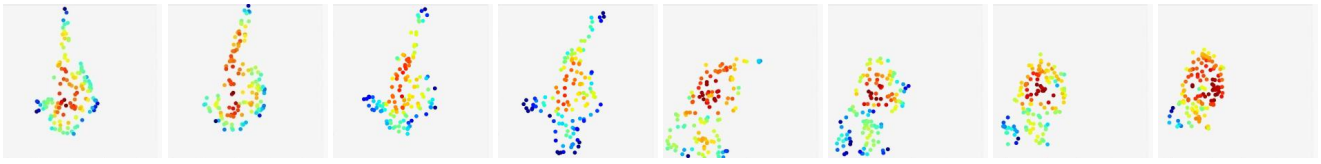
4.2. Video-level t-SNE Features

Different from PACs encoding point-level activation in intermediate layers, the last-layer features before the classifier aggregate the video-level information in the whole model. To qualitatively analyze the video-level representations, we project these last-layer features to the 2-dimensional space through t-SNE [25].

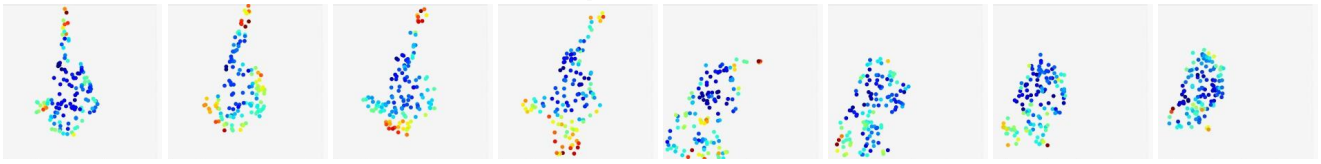
As shown in Figure 3a & 3a, in the cases without backgrounds (w/ BBox), our dynamic branch (the temporal stream) has better intra-class compactness and inter-class separability than the static one (the spatial stream). Though the dynamic branch shows overall superiority, the static stream is complementary to it: the dynamic branch sometimes confuses the orange data points with the yellow ones, while the static branch is capable of discriminating them correctly. In the case with backgrounds (w/o BBox), Figure 3c & 3d demonstrate that the background noises have negative impacts on both static and kinematic representations. In this case, our dynamic features are still highly robust to backgrounds compared with the static ones.



(a) Depth Maps, w/ BBox

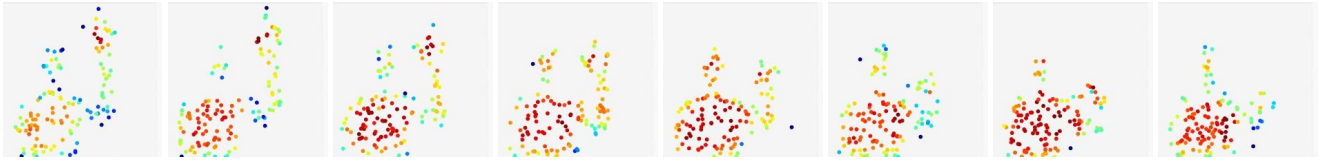


(b) Spatial (Static), w/ BBox

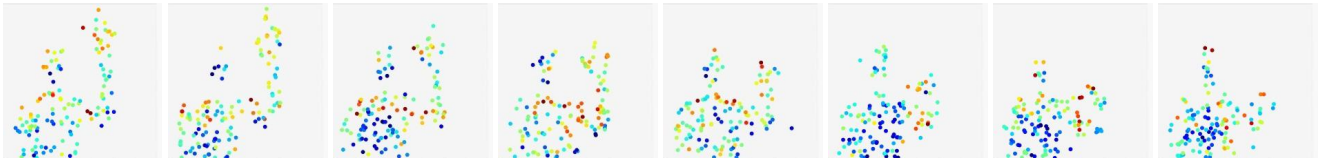


(c) Temporal (Dynamic), w/ BBox

(d) Depth Maps, w/o BBox

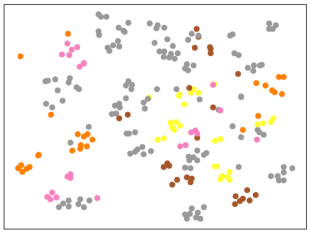


(e) Spatial (Static), w/o BBox

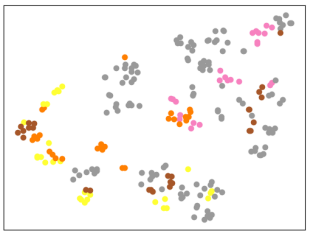


(f) Temporal (Dynamic), w/o BBox

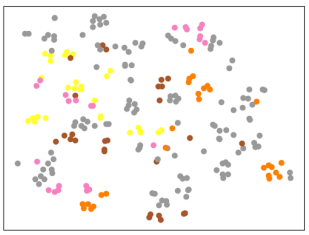
Figure 2. *Sequential raw depth inputs and point-level PACs on* SHREC'17. In PACs, the points in red have the highest activation values, while the blue ones are the lowest activating points. The animations of the above figures can be found in Figure 6 in the main body of our paper.



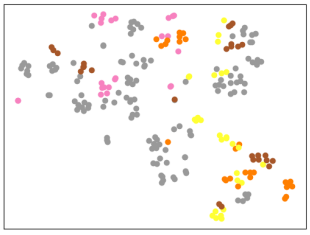
(a) Spatial (Static), w/ BBox



(b) Temporal (Dynamic), w/ BBox

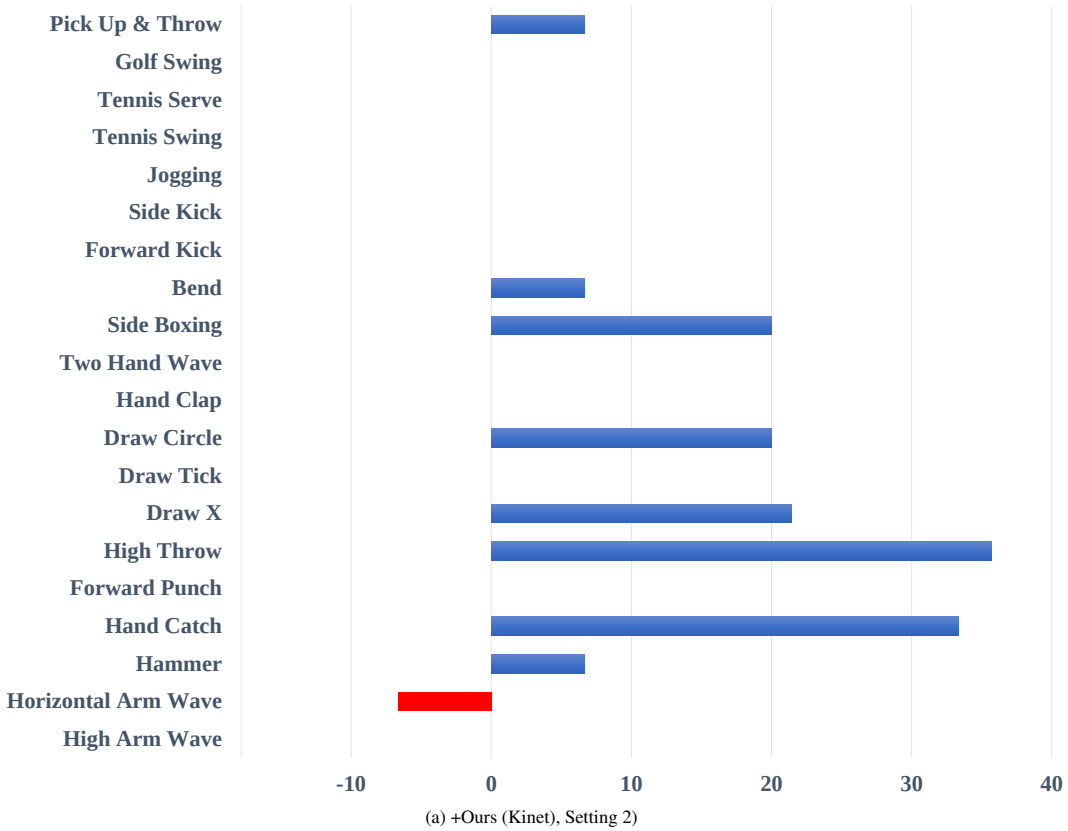


(c) Spatial (Static), w/o BBox

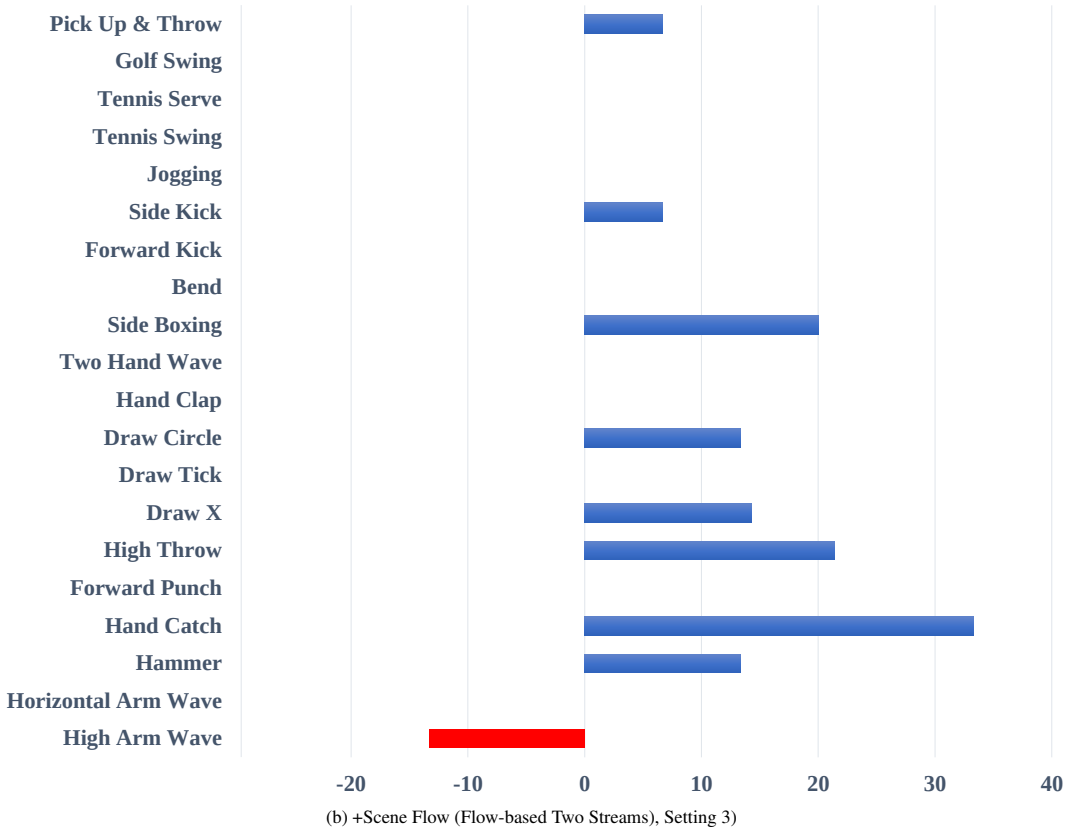


(d) Temporal (Dynamic), w/o BBox

Figure 3. *Feature visualizations on* SHREC'17 *with video-level t-SNE.* For clarity, 8 out of the 28 classes are presented in the above figures. A video is visualized as a data point, of which the same color means the identical ground-truth category. The intra-class compactness and the inter-class separability reflect the representative ability of a model.

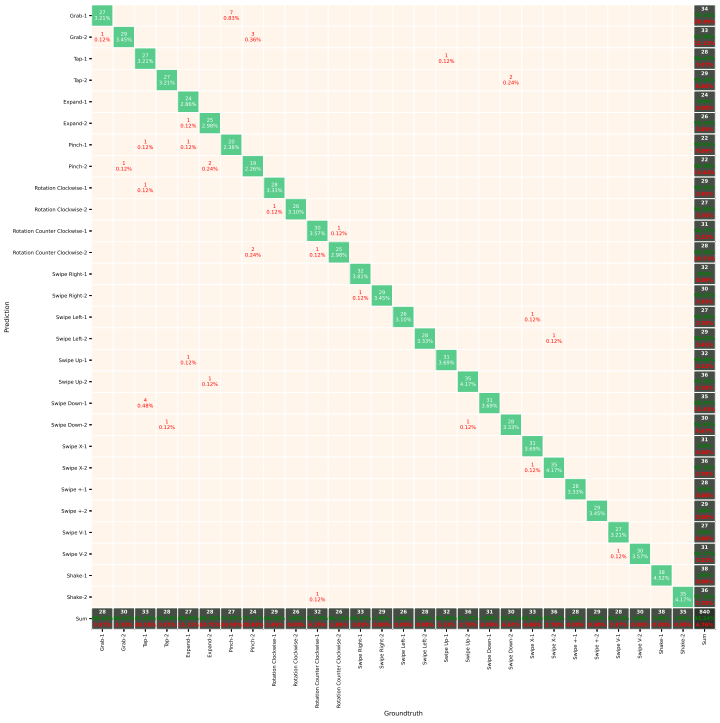


(a) +Ours (Kinet), Setting 2)

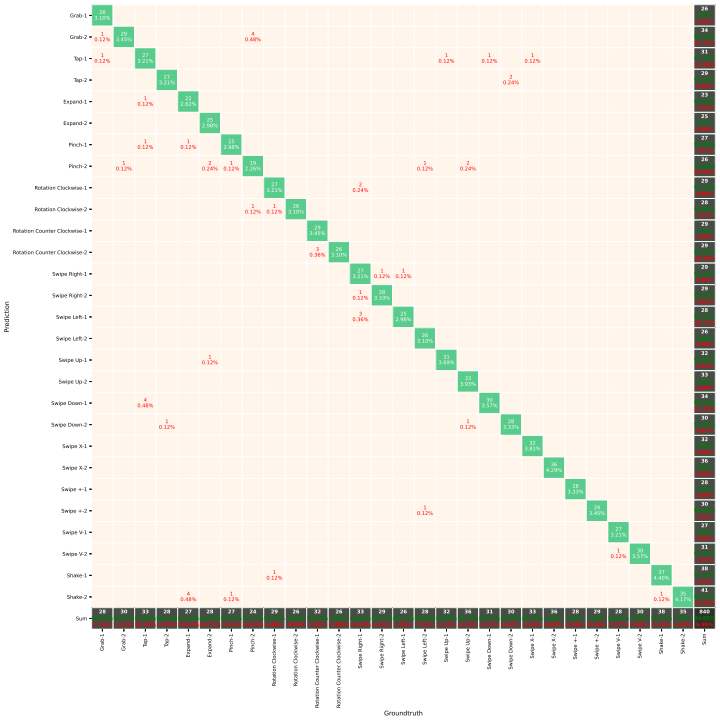


(b) +Scene Flow (Flow-based Two Streams), Setting 3)

Figure 4. *Per-class accuracy gains (%) over the static model on 16-frame* MSRAction-3D. The blue bar indicates the postive change, whereas the red one represents the negative change.



(a) w/ BBox



(b) w/o BBox

Figure 5. *Confusion matrices on* SHREC'17.

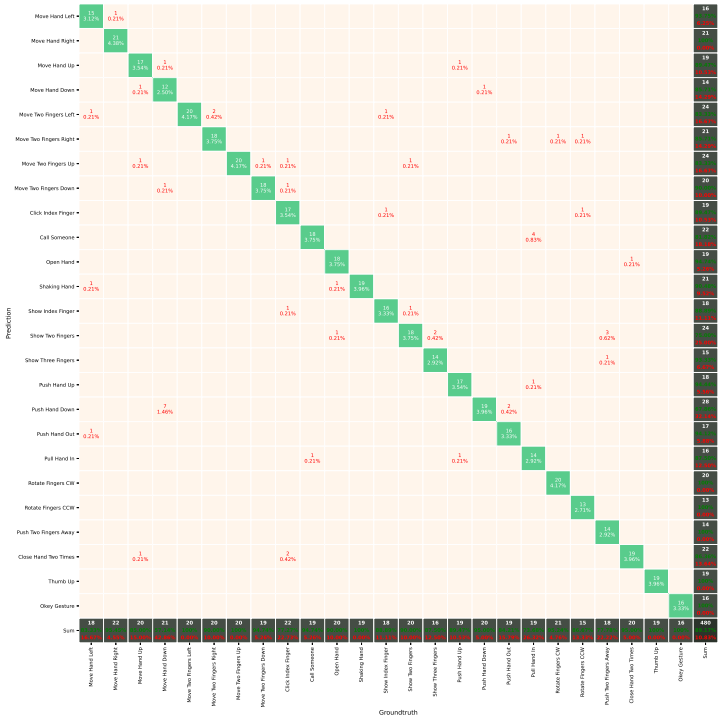


Figure 6. *Confusion matrix on* NvGesture.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 2

[2] Carlos Caetano, Jessica Sena, François Brémond, Jefersson A Dos Santos, and William Robson Schwartz. Skelemotion: A new representation of skeleton joint sequences based on motion information for 3d action recognition. In *AVSS*, 2019. 1

[3] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4D transformer networks for spatio-temporal modeling in point cloud videos. In *CVPR*, 2021. 1, 3, 5

[4] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. PSTNet: Point spatio-temporal convolution on point cloud sequences. In *ICLR*, 2021. 1, 3, 5

[5] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2, 3

[6] Aravindh Krishnamoorthy and Deepak Menon. Matrix inversion using cholesky decomposition. In *2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*, pages 70–72. IEEE, 2013. 2

[7] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Unsupervised learning of view-invariant action representations. In *NeurIPS*, 2018. 1

[8] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *CVPR*, 2019. 1

[9] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2684–2701, 2019. 1

[10] Jun Liu, Gang Wang, Ling-Yu Duan, Kamila Abdiyeva, and Alex C Kot. Skeleton-based human action recognition with global context-aware attention lstm networks. *IEEE Transactions on Image Processing*, 27(4):1586–1599, 2017. 1

[11] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. Global context-aware attention lstm networks for 3d action recognition. In *CVPR*, 2017. 1

[12] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *ICCV*, 2019. 5

[13] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *CVPR*, 2020. 1

[14] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 3

[15] Yuecong Min, Xiujuan Chai, Lei Zhao, and Xilin Chen. Flickernet: Adaptive 3D gesture recognition from sparse point clouds. In *BMVC*, 2019. 2, 3, 5

[16] Yuecong Min, Yanxiao Zhang, Xiujuan Chai, and Xilin Chen. An efficient PointLSTM for point clouds based gesture recognition. In *CVPR*, 2020. 5

[17] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *CVPR*, 2020. 3

[18] Eshed Ohn-Bar and Mohan Trivedi. Joint angles similarities and hog2 for action recognition. In *CVPRW*, 2013. 1

[19] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *CVPR*, 2013. 1

[20] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2, 3, 5

[21] 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

[22] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *CVPR*, 2019. 1

[23] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *CVPR*, 2019. 1

[24] Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *CVPR*, 2019. 1

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 6

[26] Pichao Wang, Wanqing Li, Zhimin Gao, Chang Tang, and Philip O Ogunbona. Depth pooling based large-scale 3-d action recognition with convolutional neural networks. *IEEE Transactions on Multimedia*, 20(5):1051–1061, 2018. 1

[27] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 2019. 2, 5

[28] Yancheng Wang, Yang Xiao, Fu Xiong, Wenxiang Jiang, Zhiguo Cao, Joey Tianyi Zhou, and Junsong Yuan. 3dv: 3d dynamic voxel for action recognition in depth video. In *CVPR*, 2020. 1

[29] Yang Xiao, Jun Chen, Yancheng Wang, Zhiguo Cao, Joey Tianyi Zhou, and Xiang Bai. Action recognition for depth video using multi-view dynamic images. *Information Sciences*, 480:287–304, 2019. 1

[30] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018. 2, 5

[31] Xiaodong Yang and YingLi Tian. Super normal vector for activity recognition using depth sequences. In *CVPR*, 2014. 1

[32] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive neural networks for high performance skeleton-based human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1963–1978, 2019. 1