Temporal-Viewpoint Transportation Plan for Skeletal Few-shot Action Recognition — Supplementary Material —

Lei Wang[†,§] and Piotr Koniusz[§,†]

[†]Australian National University [§]Data61/CSIRO
[§]firstname.lastname@data61.csiro.au

Below are additional derivations, evaluations and illustrations of our method.

A Prerequisites

Euler angles [89] are defined as successive planar rotation angles around x, y, and z axes. For 3D coordinates, we have the following rotation matrices $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}, \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

As the resulting composite rotation matrix depends on the order of rotation axes, *i.e.*, $\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \neq \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x$, we also investigate the algebra of stereo projection.

Stereo projections [90]. Suppose we have a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t} = [t_x, t_y, t_z]^T$ between left/right cameras (imagine some non-existent stereo camera). Let $\mathbf{M}_l$ and $\mathbf{M}_r$ be the intrinsic matrices of the left/right cameras. Let $\mathbf{p}_l$ and $\mathbf{p}_r$ be coordinates of the left/right camera. As the origin of the right camera in the left camera coordinates is $\mathbf{t}$, we have: $\mathbf{p}_r = \mathbf{R}(\mathbf{p}_l - \mathbf{t})$ and $(\mathbf{p}_l - \mathbf{t})^T = (\mathbf{R}^T\mathbf{p}_r)^T$. The plane (polar surface) formed by all points passing through $\mathbf{t}$ can be expressed by $(\mathbf{p}_l - \mathbf{t})^T(\mathbf{p}_l \times \mathbf{t}) = 0$. Then, $\mathbf{p}_l \times \mathbf{t} = \mathbf{S}\mathbf{p}_l$ where $\mathbf{S} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$. Based on the above equations, we obtain ${\mathbf{p}_r}^T\mathbf{R}\mathbf{S}\mathbf{p}_l = 0$, and note that $\mathbf{R}\mathbf{S} = \mathbf{E}$ is the Essential Matrix, and $\mathbf{p}_r^T\mathbf{E}\mathbf{p}_l = 0$ describes the relationship for the same physical point under the left and right camera coordinate system. As $\mathbf{E}$ has no internal inf. about the camera, and $\mathbf{E}$ is based on the camera coordinates, we use a fundamental matrix $\mathbf{F}$ that describes the relationship for the same physical point under the camera pixel coordinate system. The relationship between the pixel and camera coordinates is: $\mathbf{p}^* = \mathbf{M}\mathbf{p}'$ and ${\mathbf{p}'_r}^T\mathbf{E}\mathbf{p}'_l = 0$.

Now, suppose the pixel coordinates of $\mathbf{p}'_l$ and $\mathbf{p}'_r$ in the pixel coordinate system are $\mathbf{p}^*_l$ and $\mathbf{p}^*_r$, then we can write ${\mathbf{p}^*_r}^T(\mathbf{M}_r^{-1})^T\mathbf{E}\mathbf{M}_l^{-1}\mathbf{p}^*_l = 0$, where $\mathbf{F} = (\mathbf{M}_r^{-1})^T\mathbf{E}\mathbf{M}_l^{-1}$ is the fundamental matrix. Thus, the relationship for the same point in the pixel coordinate system of the left/right camera is:

$${\mathbf{p}^*_r}^T\mathbf{F}\mathbf{p}^*_l = 0. \tag{7}$$

We treat 3D body joint coordinates as $\mathbf{p}^*_l$. Given $\mathbf{F}$ (estimation of $\mathbf{F}$ is explained in **stereo projections** of Sec. 4 in the main paper), we obtain their coordinates $\mathbf{p}^*_r$ in the new view.

GNN notations. Firstly, let $G=(\mathbf{V},\mathbf{E})$ be a graph with the vertex set $\mathbf{V}$ with nodes $\{v_1,...,v_n\}$, and $\mathbf{E}$ are edges of the graph. Let $\mathbf{A}$ and $\mathbf{D}$ be the adjacency and diagonal degree matrix, respectively. Let $\tilde{\mathbf{A}}=\mathbf{A}+\mathbf{I}$ be the adjacency matrix with self-loops (identity matrix) with the corresponding diagonal degree matrix $\tilde{\mathbf{D}}$ such that $\tilde{D}_{ii}=\sum_j(\mathbf{A}^{ij}+\mathbf{I}^{ij})$. Let $\mathbf{S}=\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ be the normalized adjacency matrix with added self-loops. For the l-th layer, we use $\mathbf{\Theta}^{(l)}$ to denote the learnt weight matrix, and $\mathbf{\Phi}$ to denote the outputs from the graph networks. Below, we list backbones used by us.

GCN [96]. GCNs learn the feature representations for the features $\mathbf{x}_i$ of each node over multiple layers. For the l-th layer, we denote the input by $\mathbf{H}^{(l-1)}$ and the output by $\mathbf{H}^{(l)}$. Let the input (initial) node representations be $\mathbf{H}^{(0)}=\mathbf{X}$. For an L-layer GCN, the output representations are given by:

$$\mathbf{\Phi}_{\text{GCN}}=\mathbf{S}\mathbf{H}^{(L-1)}\mathbf{\Theta}^{(L)} \text{ where } \mathbf{H}^{(l)}=\text{ReLU}(\mathbf{S}\mathbf{H}^{(l-1)}\mathbf{\Theta}^{(l)}). \tag{8}$$

APPNP [97]. The Personalized Propagation of Neural Predictions (PPNP) and its fast approximation, APPNP, are based on the personalized PageRank. Let $\mathbf{H}^{(0)}=f_{\mathbf{\Theta}}(X)$ be the input to APPNP, where $f_{\mathbf{\Theta}}$ can be an MLP with parameters $\mathbf{\Theta}$. Let the output of the l-th layer be $\mathbf{H}^{(l)}=(1-\alpha)\mathbf{S}\mathbf{H}^{(l-1)}+\alpha\mathbf{H}^{(0)}$, where α is the teleport (or restart) probability in range $(0,1]$. For an L-layer APPNP, we have:

$$\mathbf{\Phi}_{\text{APPNP}}=(1-\alpha)\mathbf{S}\mathbf{H}^{L}+\alpha\mathbf{H}^{(\mathbf{0})}. \tag{9}$$

SGC [107] & **S^{2}GC** [109]. SGC captures the L-hops neighborhood in the graph by the L-th power of the transition matrix used as a spectral filter. For an L-layer SGC, we obtain:

$$\mathbf{\Phi}_{\text{SGC}}=\mathbf{S}^{L}\mathbf{X}\mathbf{\Theta}. \tag{10}$$

Based on a modified Markov Diffusion Kernel, Simple Spectral Graph Convolution (S^{2}GC) is the summation over l-hops, $l=1,...,L$. The output of S^{2}GC is:

$$\mathbf{\Phi}_{\text{S}^2\text{GC}}=\frac{1}{L}\sum_{l=1}^{L}((1-\alpha)\mathbf{S}^{l}\mathbf{X}+\alpha\mathbf{X})\mathbf{\Theta}. \tag{11}$$

Soft-DTW [91,92]. Dynamic Time Warping can be seen as a specialized case of the Wasserstein metric, under specific transportation plan. Soft-DTW is defined as:

$$d_{\text{DTW}}(\mathbf{\Psi},\mathbf{\Psi}')=\underset{\mathbf{A}\in\mathcal{A}_{\tau,\tau'}}{\text{SoftMin}_{\gamma}}\left\langle\mathbf{A},\boldsymbol{D}(\mathbf{\Psi},\mathbf{\Psi}')\right\rangle, \tag{12}$$

$$\text{where } \text{SoftMin}_{\gamma}(\boldsymbol{\alpha})=-\gamma\log\sum_{i}\exp(-\alpha_i/\gamma). \tag{13}$$

The binary $\mathbf{A}\in\mathcal{A}_{\tau,\tau'}$ denotes a path within the transportation plan $\mathcal{A}_{\tau,\tau'}$ which depends on lengths τ and τ' of sequences $\mathbf{\Psi}\equiv[\boldsymbol{\psi}_1,...,\boldsymbol{\psi}_\tau]\in\mathbb{R}^{d'\times\tau}$, $\mathbf{\Psi}'\equiv[\boldsymbol{\psi}'_1,...,\boldsymbol{\psi}'_{\tau'}]\in\mathbb{R}^{d'\times\tau'}$ and $\boldsymbol{D}\in\mathbb{R}_{+}^{\tau\times\tau'}\equiv[d_{\text{base}}(\boldsymbol{\psi}_m,\boldsymbol{\psi}'_n)]_{(m,n)\in\mathcal{I}_\tau\times\mathcal{I}_{\tau'}}$, the matrix of distances, is evaluated for $\tau\times\tau'$ frame representations according to some base distance $d_{\text{base}}(\cdot,\cdot)$, *i.e.*, the Euclidean or the RBF-induced distance. We make use of principles of soft-DTW. However, we design a joint alignment between temporal skeleton sequences and simulated skeleton viewpoints, an entirely novel proposal.

Table 9: Seven publicly available benchmark datasets which we use for FSAR.

Datasets		Year	Classes	Subjects	#views	#clips	Sensor	Modalities	#joints
MSRAction3D	[98]	2010	20	10	1	567	Kinect v1	Depth + 3DJoints	20
3D Action Pairs	[100]	2013	12	10	1	360	Kinect v1	RGB + Depth + 3DJoints	20
UWA3D Activity	[101]	2014	30	10	1	701	Kinect v1	RGB + Depth + 3DJoints	15
UWA3D Multiview Activity II	[102]	2015	30	9	4	1,070	Kinect v1	RGB + Depth + 3DJoints	15
NTU RGB+D	[103]	2016	60	40	80	56,880	Kinect v2	RGB + Depth + IR + 3DJoints	25
NTU RGB+D 120	[99]	2019	120	106	155	114,480	Kinect v2	RGB + Depth + IR + 3DJoints	25
Kinetics-skeleton	[108]	2018	400	-	-	$\sim$ 300,000	-	RGB + 2DJoints	18

Table 10: Evaluations of backbones on 5 datasets.

	MSRAction3D		3DAct.Pairs	UWA3DActivity		NTU-60	NTU-120
	5-way	10-way	5-way	5-way	10-way	50-way	20-way
GCN	56.0±1.3	37.6±1.2	-	55.4±0.8	42.4±0.8	56.0	-
SGC	66.0±1.1	48.3±1.1	69.0±1.8	56.4±0.7	41.6±0.6	68.1	30.7
APPNP	67.2±0.8	58.1±0.8	69.0±2.0	60.6±1.5	42.4±1.3	68.5	30.8
S^2GC (Eucl.)	68.8±1.2	63.1±0.9	72.2±1.8	69.8±0.7	58.3±0.6	75.6	34.5
S^2GC (RBF)	**73.2±0.9**	**64.6±0.8**	**75.6±2.1**	**76.4±0.7**	**58.9±0.7**	**78.1**	**36.2**

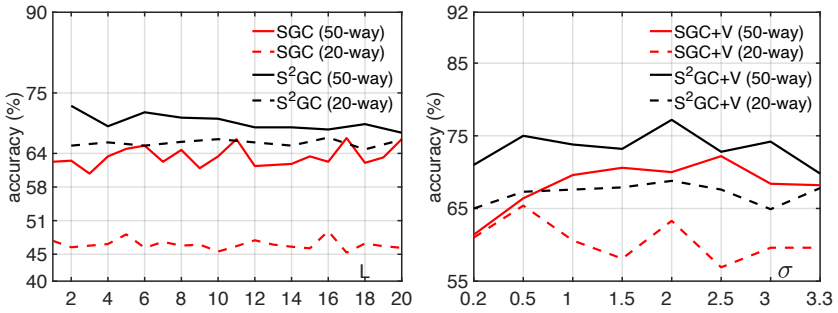
B Datasets and their statistics

Table 9 contains statistics of datasets used in our experiments. Smaller datasets below are used for the backbone selection and ablations:

- *MSRAction3D* [98] is an older AR datasets captured with the Kinect depth camera. It contains 20 human sport-related activities such as *jogging*, *golf swing* and *side boxing*.
- *3D Action Pairs* [100] contains 6 selected pairs of actions that have very similar motion trajectories, *e.g.*, *put on a hat* and *take off a hat*, *pick up a box* and *put down a box*, *etc*.
- *UWA3D Activity* [101] has 30 actions performed by 10 people of various height at different speeds in cluttered scenes.

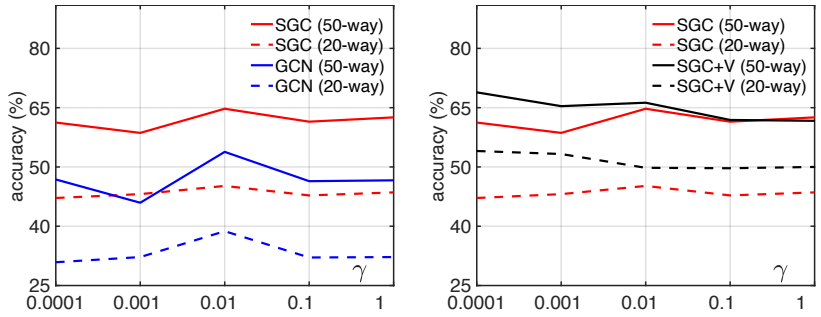
As MSRAction3D, 3D Action Pairs, and UWA3D Activity have not been used in FSAR, we created 10 training/testing splits, by choosing half of class concepts for training, and half for testing per split per dataset. Training splits were further subdivided for crossvalidation.

Sec. G.1 details the class concepts per split for small datasets.



(a) (b)

Fig. 7: Evaluations of L and σ. (a): L for SGC and S^{2}GC. (b): σ of RBF distance for Eq. (2) (SGC and S^{2}GC, NTU-60).



(a) (b)

Fig. 8: Evaluations w.r.t. γ. (a): γ in Eq. (2) with the temporal alignment alone. (b): comparisons of temporal alignment alone *vs*. temporal-viewpoint alignment (*V*) on NTU-60.

C Backbone selection and hyperparameter evaluation

C.1 Backbone selection

We conduct experiments on 4 GNN backbones listed in Table 10. S^{2}GC performs the best on all datasets including large-scale NTU-60 and NTU-120, APPNP outperforms SGC, and SGC outperforms GCN. We note that using the RBF-induced distance for $d_{base}(\cdot,\cdot)$ of DTW outperforms the Euclidean distance. Fig. 7 shows a comparison of using SGC and S^{2}GC on NTU-60. As shown in the figure that using S^{2}GC performs better than SGC for both 50-way and 20-way settings. We also notice that results w.r.t. the number of layers L are more stable for S^{2}GC than SGC. We choose $L = 6$ for our experiments. Fig. 7b shows evaluations of σ for the RBF-induced distance for both SGC and S^{2}GC. As $\sigma = 2$ in S^{2}GC achieves the highest performance (both 50-way and 20-way), we choose S^{2}GC as the backbone and $\sigma = 2$ for the experiments.

Table 11 is a comparison of CNN, RNN and GNN as backbones of JEANIE. The role of this backbone in JEANIE is to process the per-block per-viewpoint body joint

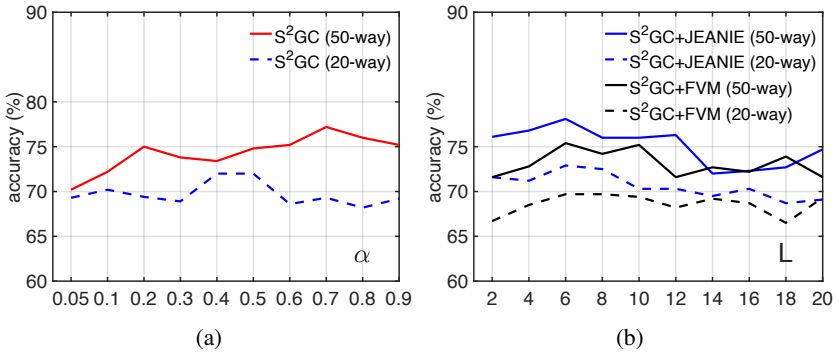


Fig. 9: Evaluations of (a) α and (b) the number of layers L for S^{2}GC on NTU-60.

features obtained from MLP and exploit interactions among body joints. The input and output feature dimension (per temporal block) of backbone (*e.g.*, CNN, RNN and GNN) are $J \times d$. For CNN, we simply use 2D convolution (square kernels and equal stride with auto padding to ensure the input and output feature maps have the same dimensions). For RNN (joint-wise), we use J RNN cells, each RNN processes d-dimensional vector for each body joint (J-input to J-output). Both input and output are d-dimensional feature vectors. The outputs from J RNN cells are concatenated to form $J \times d$ feature maps. For RNN (temporal-wise), we modify the first MLP in our pipeline, to produce $J \times d$ vector per each time-step $1, ..., T$ of temporal block. We use T RNN cells, each RNN processes $J \times d$-dimensional vector (T-input to 1-output). The output we use is from the last of T RNN cells is $J \times d$. As shown in the table, GNN outperforms RNN, and RNN outperforms CNN. Note that our GNN (S^{2}GC) is a simple linear projection on a spectral filter of graph and without learnable parameters (Eq. (11) of Supplementary Material). We believe RNN (temporal-wise) is better than RNN (joint-wise) as joints to not have well defined time-like order (thus RNN is bad) while GNN uses the graph connectivity (topological order).

Table 11: A comparison of different backbones in JEANIE on NTU-60 (#training classes = 10).

Backbones	CNN	RNN (joint-wise)	RNN (temporal-wise)	GNN (ours)
Results (acc.)	55.3	59.9	61.1	**65.0**

C.2 Evaluations of viewpoint alignment

Fig. 8 shows comparisons with temporal-viewpoint alignment (*V*) *vs*. temporal alignment alone on NTU-60.

Table 12: Experimental results of stride for degrees on NTU-60.

	10	20	30	40	50
5°	63.8	73.7	74.2	75.0	76.5
10°	64.2	74.8	75.0	78.0	79.1
15°	**65.0**	**75.2**	**76.7**	**78.9**	**80.0**
30°	**65.0**	74.8	75.0	76.9	78.5
45°	60.0	68.5	71.0	71.5	72.0

Fig. 8a shows evaluations of γ without the viewpoint alignment. Fig. 8b shows that temporal-viewpoint alignment (*V*) brings around 5% (20- and 50-way protocols, $\gamma = 0.0001$) improvement.

C.3 Evaluations w.r.t. α

Figure 9a shows the evaluations of α for the S^{2}GC backbone. As shown in the plot, for 50-way protocol, the best performance is achieved when $\alpha = 0.7$ ($\alpha = 0.5$ is the second best performer for the 50-way protocol)). For the 20-way protocol, the top performer is $\alpha = 0.4$ or $\alpha = 0.5$. Thus, we chose $\alpha = 0.5$ in our experiments. Please note we observed the same trend on the validation split.

C.4 Evaluations w.r.t. the number of layers L

Figure 9b shows the performance w.r.t. the number of layers L used by S^{2}GC and S^{2}GC+JEANIE. As shown in this plot, when $L = 6$, S^{2}GC with JEANIE performs the best for both 20- and 50-way experiments. For the Free Viewpoint Matching (FVM) using S^{2}GC (S^{2}GC+FVM), the performance is not as stable as in the case of S^{2}GC+JEANIE.

C.5 Evaluation of stride for viewing angles

The stride for viewing angles is a mere equivalent of stride parameter in CNNs, which is an equal interval location sampler. We show various sampling steps for viewing angles in Table 12. We notice that when stride is 15°, JEANIE performs the best on NTU-60.

D Inference Time

Table 13 below compares training and inference times per query on Titan RTX 2090. For soft-DTW, each query is augmented by $K \times K' = 9$ viewpoints. In the test time, we average match distance over $K \times K' = 9$ viewpoints of each test query (this is a popular standard test augmentation strategy) w.r.t. support samples. This strategy is denoted as soft-DTW$_{\text{aug}}$. We also apply the above strategy to TAP (denoted as TAP$_{\text{aug}}$). JEANIE also uses $K \times K' = 9$ viewpoints per query. We exclude the time of applying viewpoint

Table 13: A comparison of training/inference time (per query) on NTU-60 (#training classes = 10).

	Training time (s)	Inference time (s)	Total inference time (s)	Acc. (%)
soft-DTW$_{\text{aug}}$	0.098	0.019	178.5	56.8
TAP$_{\text{aug}}$	0.124	0.024	225.5	57.6
JEANIE	0.099	0.020	187.0	65.0

Table 14: Evaluations of additional baselines on NTU-60.

# Training Classes	10	20	30	40	50
Matching Nets [106] (skeleton to image tensor)	26.7	30.6	32.9	36.4	39.9
Proto. Net [104] (skeleton to image tensor)	30.6	33.9	36.8	40.2	43.0
Proto. Net (per block image tensor, temp. align.)	40.4	42.4	45.2	49.0	50.3
Proto. Net (per block image tensor, temp. & view. align.)	41.6	43.0	47.7	50.4	51.6

generation as skeletons can be pre-processed at once (1.6h with non-optimized CPU code) and stored for the future use. Among methods which use multiple viewpoints, JEANIE outperforms soft-DTW$_{\text{aug}}$ and TAP$_{\text{aug}}$ by 8.2% and 7.4% respectively. JEANIE outperforms ordinary soft-DTW and TAP by 11.3% and 10.8%. For soft-DTW$_{\text{aug}}$ and TAP$_{\text{aug}}$, their total training and testing were about 5× and 9× slowed compared to counterpart soft-DTW and TAP. This is expected as they had to deal with $K \times K' = 9$ more samples. We tried also parallel JEANIE. Training JEANIE$_{\text{par}}$ with 4 Titan RTX 2090 took 44h, the total inference was 48s.

E More baselines on NTU-60

Table 14 shows more evaluations on NTU-60. Before GCNs have become mainstream backbones for the 3D Skeleton-based Action Recognition, encoding 3D body joints of skeletons as texture-like images enjoyed some limited popularity, with approaches [94,95,105] feeding such images into CNN backbones. This facilitates easy FSL with existing pipelines such as Matching Nets [106] and Prototypical Net [104]. Thus, we reshape the normalized 3D coordinates of each skeleton sequence or per block skeleton into image tensors, and pass them into Matching Nets [106] and Prototypical Net [104] for few-shot learning. Not surprisingly, using texture-like images for skeletons is sub-optimal. Our JEANIE is more than 25% better.

F Drawbacks/Limitations

Some minor drawbacks of our work are: (i) extending our work to video-based action recognition requires more work as it is hard to simulate the views of pixels for RGB

videos; (ii) alignment in 4D space is slower than in 2D space and slower than no alignment at all, but improvements in accuracy are significant, and in fact the alignment step can be written as the RKHS kernel, which can be linearized by the Nyström feature maps, casting 4D problem as two 2D problems.

G Evaluation Protocols

Below, we detail our new/additional evaluation protocols used in the experiments.

G.1 Few-shot AR protocols on the small-scale datasets

Below, we explain the selection process.

FSAR (MSRAction3D) . As this dataset contains 20 action classes, we randomly choose 10 action classes for training and the rest 10 for testing. We repeat this sampling process 10 times to form in total 10 train/test splits. For each split, we have 5-way and 10-way experimental settings. The overall performance on this dataset is computed by averaging the performance on 10 splits.

FSAR (3D Action Pairs) . This dataset has in total 6 action pairs (12 action classes), each pair of action has very similar motion trajectories, *e.g.*, *pick up a box* and *put down a box*. We randomly choose 3 action pairs to form a training set (6 action classes) and the half action pairs for the test set, and in total there are $\binom{n}{k} = \binom{6}{3} = 20$ different combinations of train/test splits. As our train/test splits are based on action pairs, we are able to test whether the algorithm is able to classify unseen action pairs that share similar motion trajectories. We use 5-way protocol on this dataset to evaluate the performance of FSAR, averaged over all 20 splits.

FSAR (UWA3D Activity) . This dataset has 30 action classes. We randomly choose 15 action classes for training and the rest half action classes for testing. We form in total 10 train/test splits, and we use 5-way and 10-way protocols on this dataset, averaged over all 10 splits.

G.2 One-shot protocol on NTU-60

Following NTU-120 [99], we introduce the one-shot AR setting on NTU-60. We split the whole dataset into two parts: auxiliary set (on NTU-120 the training set is called as auxiliary set, so we follow such a terminology) and one-shot evaluation set.

Auxiliary set contains 50 classes, and all samples of these classes can be used for learning and validation. Evaluation set consists of 10 novel classes, and one sample from each novel class is picked as the exemplar (terminology introduced by authors of NTU-120), while all the remaining samples of these classes are used to test the recognition performance.

Evaluation set contains 10 novel classes, namely, A1, A7, A13, A19, A25, A31, A37, A43, A49, A55. The following 10 samples are the exemplars:
(01)S001C003P008R001A001, (02)S001C003P008R001A007,
(03)S001C003P008R001A013, (04)S001C003P008R001A019,
(05)S001C003P008R001A025, (06)S001C003P008R001A031,
(07)S001C003P008R001A037, (08)S001C003P008R001A043,
(09)S001C003P008R001A049, (10)S001C003P008R001A055.

Auxiliary set contains 50 classes (the remaining 50 classes of NTU-60 excluding the 10 classes in evaluation set).

G.3 Few-shot multiview classification on NTU-120

Horizontal camera view . As NTU-120 is captured by 3 cameras (from 3 different horizontal angles: -45°, 0°, 45°), we split the whole dataset based on the camera ID to form our 3 horizontal camera viewpoints (left, center and right views). We then evaluate few-shot multiview classification using (i) the left view for training and the center view for testing (ii) the left view for training and the right view for testing (ii) the left and center views for training and the right view for testing.

Vertical camera view . Based on the table provided in [99], we first group 32 camera setups into 3 groups by dividing the range of heights into 3 equally-sized ranges to form roughly the top, center and bottom views. We then group the whole dataset into 3 camera viewpoints based on the camera setup IDs. For few-shot multiview classification, we evaluate our proposed method using (i) bottom view for training and center view for testing (ii) bottom view for training and top view for testing (iii) bottom and center views for training and top view for testing.

H Network configuration and training details

Below we provide the details of network configuration and training process in the following sections.

H.1 Network configuration

Given the temporal block size M (the number of frames in a block) and desired output size d, the configuration of the 3-layer MLP unit is: FC ($3M \rightarrow 6M$), LayerNorm (LN) as in [93], ReLU, FC ($6M \rightarrow 9M$), LN, ReLU, Dropout (for smaller datasets, the dropout rate is 0.5; for large-scale datasets, the dropout rate is 0.1), FC ($9M \rightarrow d$), LN. Note that M is the temporal block size and d is the output feature dimension per body joint. Note that ablations on the value of M are already conducted in Table 3.

Backbone with GNN and Transformer . Following EN described in Section 3, let us take the query input $\mathbf{X} \in \mathbb{R}^{3\times J\times M}$ for the temporal block of length M as an example, where 3 indicates 3D Cartesian coordinate and J is the number of body joints. As alluded to earlier, we obtain $\widehat{\mathbf{X}}^T = \text{MLP}(\mathbf{X}; \mathcal{F}_{MLP}) \in \mathbb{R}^{d\times J}$.

Subsequently, we employ a GNN and the transformer encoder [93] which consists of alternating layers of Multi-Head Self-Attention (MHSA) and a feed-forward MLP (two FC layers with a GELU non-linearity between them). LayerNorm (LN) is applied before every block, and residual connections after every block. Each block feature matrix $\widehat{\mathbf{X}} \in \mathbb{R}^{J \times d}$ encoded by GNN (without learnable $\mathbf{\Theta}$) is then passed to the transformer. Similarly to the standard transformer, we prepend a learnable vector $\mathbf{y}_{\text{token}} \in \mathbb{R}^{1 \times d}$ to the sequence of block features $\widehat{\mathbf{X}}$ obtained from GNN, and we also add the positional embeddings $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(1+J) \times d}$ based on the sine and cosine functions (standard in transformers) so that token $\mathbf{y}_{\text{token}}$ and each body joint enjoy their own unique positional encoding. We obtain $\mathbf{Z}_0 \in \mathbb{R}^{(1+J) \times d}$ which is the input in the following backbone:

$$\mathbf{Z}_0 = [\mathbf{y}_{\text{token}}; \text{GNN}(\widehat{\mathbf{X}})] + \mathbf{E}_{\text{pos}}, \tag{14}$$

$$\mathbf{Z}'_k = \text{MHSA}(\text{LN}(\mathbf{Z}_{k-1})) + \mathbf{Z}_{k-1}, \;\; k = 1, ..., L_{\text{tr}} \tag{15}$$

$$\mathbf{Z}_k = \text{MLP}(\text{LN}(\mathbf{Z}'_k)) + \mathbf{Z}'_k, \qquad k = 1, ..., L_{\text{tr}} \tag{16}$$

$$\mathbf{y}' = \text{LN}\big(\mathbf{Z}^{(0)}_{L_{\text{tr}}}\big) \qquad \text{where} \qquad \mathbf{y}' \in \mathbb{R}^{1 \times d} \tag{17}$$

$$f(\mathbf{X}; \mathcal{F}) = \text{FC}(\mathbf{y}'^T; \mathcal{F}_{FC}) \qquad \in \mathbb{R}^{d'}, \tag{18}$$

where $\mathbf{Z}^{(0)}_{L_{\text{tr}}}$ is the first d dimensional row vector extracted from the output matrix $\mathbf{Z}_{L_{\text{tr}}}$ of size $(J+1) \times d$ which corresponds to the last layer L_{tr} of the transformer. Moreover, parameter L_{tr} controls the depth of the transformer, whereas $\mathcal{F} \equiv [\mathcal{F}_{MLP}, \mathcal{F}_{GNN}, \mathcal{F}_{Transf}, \mathcal{F}_{FC}]$ is the set of parameters of EN. In case of APPNP, SGC and S^{2}GC, $|\mathcal{F}_{GNN}| = 0$ because we do not use their learnable parameters $\mathbf{\Theta}$ (*i.e.*, think $\mathbf{\Theta}$ is set as the identity matrix).

As in Section 3, one can define now a support feature map as $\boldsymbol{\Psi}' = [f(\boldsymbol{X}_1; \mathcal{F}), ..., f(\boldsymbol{X}_{\tau'}; \mathcal{F})] \in \mathbb{R}^{d' \times \tau'}$ for τ' temporal blocks, and the query map $\boldsymbol{\Psi}$ accordingly.

The hidden size of our transformer (the output size of the first FC layer of the MLP in Eq. (16)) depends on the dataset. For smaller datasets, the depth of the transformer is $L_{\text{tr}} = 6$ with 64 as the hidden size, and the MLP output size is $d = 32$ (note that the MLP which provides $\widehat{\mathbf{X}}$ and the MLP in the transformer must both have the same output size). For NTU-60, the depth of the transformer is $L_{\text{tr}} = 6$, the hidden size is 128 and the MLP output size is $d = 64$. For NTU-120, the depth of the transformer is $L_{\text{tr}} = 6$, the hidden size is 256 and the MLP size is $d = 128$. For Kinetics-skeleton, the depth for the transformer is $L_{\text{tr}} = 12$, hidden size is 512 and the MLP output size is $d = 256$. The number of Heads for the transformer of smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton is set as 6, 12, 12 and 12, respectively.

The output sizes d' of the final FC layer in Eq. (18) are 50, 100, 200, and 500 for the smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton, respectively.

H.2 Training details

The weights for the pipeline are initialized with the normal distr. (zero mean and unit standard dev.). We use 1e-3 for the learning rate, and the weight decay is 1e-6. We use

the SGD optimizer. We set the number of training episodes to 100K for NTU-60, 200K for NTU-120, 500K for 3D Kinetics-skeleton, 10K for small datasets such as UWA3D Multiview Activity II. We use Hyperopt for hyperparam. search on validation sets for all the datasets.

I Skeleton Data Preprocessing

Before passing the skeleton sequences into MLP and graph networks (*e.g.*, S^{2}GC), we first normalize each body joint w.r.t. to the torso joint $\mathbf{v}_{f,c}$:

$$\mathbf{v}'_{f,i} = \mathbf{v}_{f,i} - \mathbf{v}_{f,c}, \tag{19}$$

where f and i are the index of video frame and human body joint respectively. After that, we further normalize each joint coordinate into [-1, 1] range:

$$\hat{\mathbf{v}}_{f,i}[j] = \frac{\mathbf{v}'_{f,i}[j]}{\max([\text{abs}(\mathbf{v}'_{f,i}[j])]_{f\in\mathcal{I}_\tau, i\in\mathcal{I}_J})}, \tag{20}$$

where j is for selection of the x, y and z axes, τ is the number of frames and J is the number of 3D body joints per frame.

For the skeleton sequences that have more than one performing subject, (i) we normalize each skeleton separately, and each skeleton is passed to MLP for learning the temporal dynamics, and (ii) for the output features per skeleton from MLP, we pass them separately to graph networks, *e.g.*, two skeletons from a given video sequence will have two outputs from the graph networks, and we aggregate the outputs through average pooling before passing to FVM or JEANIE.

J Additional Visualizations

To explain what makes JEANIE perform well on the task of comparing pairs of sequences, we perform additional visualisations.

To this end, we choose skeleton sequences from UWA3D Multiview Activity II for experiments and visualizations of FVM and JEANIE. UWA3D Multiview Activity II contains rich viewpoint configurations and so is perfect for our investigations.

1. Matching similar actions. We choose a *walking* skeleton sequence (‘`a12_s01_e01_v01`’) as the query sample with more viewing angles for the camera viewpoint simulation, and we select another *walking* skeleton sequence of a different view (‘`a12_s01_e01_v03`’) and a *running* skeleton sequence (‘`a20_s01_e01_v02`’) as support samples respectively, to verify that our JEANIE is able to find the better matching distances compared to FVM.

2. Matching actions with similar motion trajectories. We choose a *two hand punching* skeleton sequence (‘`a04_s01_e01_v01`’) as the query sample with more viewing angles for the camera viewpoint simulation, and we select another *two hand punching* skeleton sequence of a different view (‘`a04_s05_e01_v02`’) and a *holding head* skeleton sequence (‘`a10_s05_e01_v02`’) as support samples respectively, to verify that our JEANIE is able to find the better matching distances compared to FVM.

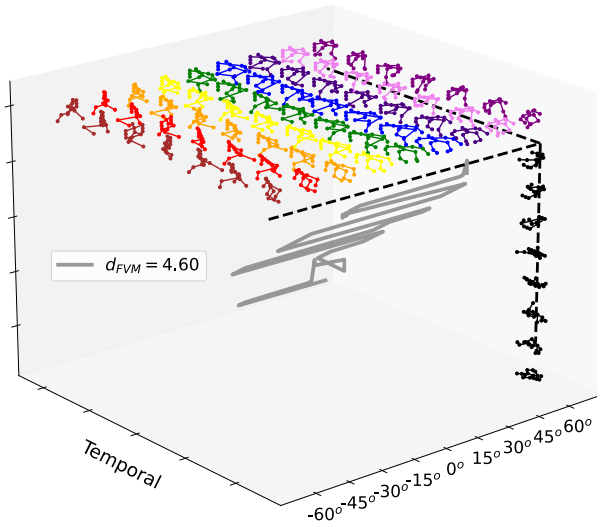
Fig. 10 and 11 show the visualizations. Comparing Fig. 10a and 10b of FVM, we notice that for skeleton sequences from different action classes (*walking vs.running*), FVM finds the path with a very small distance $d_{\text{FVM}} = 2.68$. In contrast, for sequences from the same action class (*walking vs.walking*), FVM gives $d_{\text{FVM}} = 4.60$ which higher than in case of within-class sequences. This is an undesired effect which may result in wrong comparison decision.

In contrast, in Fig 11a and 11b, our JEANIE gives $d_{\text{JEANIE}} = 8.57$ for sequences of the same action class and $d_{\text{JEANIE}} = 11.21$ for sequences from different action classes, which means that the within-class distances are smaller than between-class distances. This is in fact a very important property when comparing pairs of sequences.

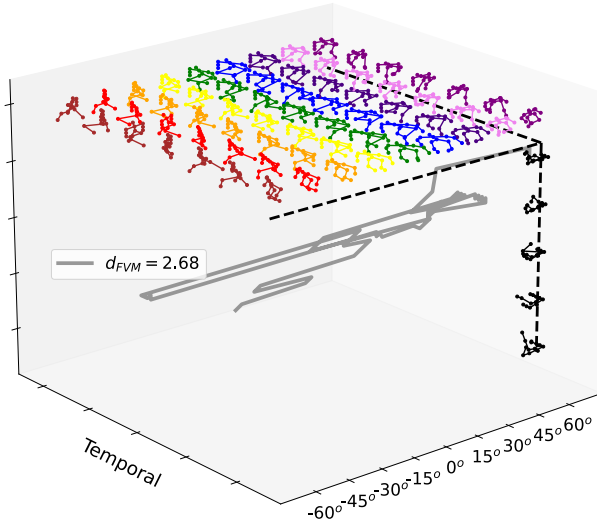
Fig 12 and 13 show additional visualizations. Again, our JEANIE produces more reasonable matching distances than FVM.

References

89. Euler angles. Wikipedia, https://en.wikipedia.org/wiki/Euler_angles, accessed: 08-03-2022
90. Lecture 12: Camera projection. On-line, http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf, accessed: 08-03-2022
91. Cuturi, M.: Fast global alignment kernels. In: ICML (2011)
92. Cuturi, M., Blondel, M.: Soft-dtw: a differentiable loss function for time-series. In: ICML (2017)
93. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020)
94. Ke, Q., Bennamoun, M., An, S., Sohel, F., Boussaid, F.: A new representation of skeleton sequences for 3d action recognition. In: CVPR (2017)
95. Ke, Q., Bennamoun, M., An, S., Sohel, F., Boussaid, F.: Learning clip representations for skeleton-based 3d action recognition. IEEE TIP **27**(6), 2842–2855 (2018)
96. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
97. Klicpera, J., Bojchevski, A., Gunnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. In: ICLR (2019)
98. Li, W., Zhang, Z., Liu, Z.: Action Recognition Based on A Bag of 3D Points. In: CVPR. pp. 9–14 (2010)
99. Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L.Y., Kot, A.C.: Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. IEEE TPAMI (2019)
100. Oreifej, O., Liu, Z.: HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In: CVPR. pp. 716–723 (2013)
101. Rahmani, H., Mahmood, A., Huynh, D.Q., Mian, A.: HOPC: Histogram of Oriented Principal Components of 3D Pointclouds for Action Recognition. In: ECCV. pp. 742–757 (2014)
102. Rahmani, H., Mahmood, A., Huynh, D.Q., Mian, A.: Histogram of Oriented Principal Components for Cross-View Action Recognition. IEEE TPAMI pp. 2430–2443 (2016)
103. Shahroudy, A., Liu, J., Ng, T.T., Wang, G.: Ntu rgb+d: A large scale dataset for 3d human activity analysis. In: CVPR (2016)
104. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) NeurIPS. pp. 4077–4087 (2017)

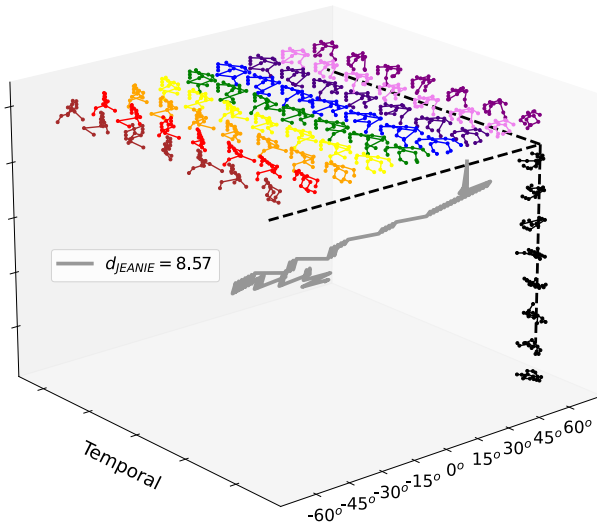


(a) *walking vs. walking* ($d_{\text{FVM}} = 4.60$)

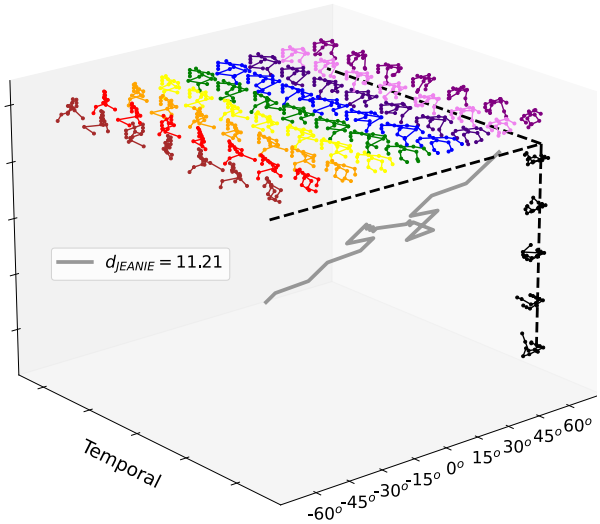


(b) *walking vs. running* ($d_{\text{FVM}} = 2.68$)

Fig. 10: Visualization of FVM for *walking vs. walking* (two different sequences) and *walking vs. running*. From UWA3D Multiview Activity II, we choose a *walking* sequence as the query sample (‘`a12_s01_e01_v01`’). We choose another *walking* sequence from a different view (‘`a12_s01_e01_v03`’) and a *running* sequence (‘`a20_s01_e01_v02`’) as the support samples respectively. We notice that for two different action sequences in (b), the greedy FVM finds the path with a very small distance $d_{\text{FVM}} = 2.68$ but for sequences of the same action class, FVM gives $d_{\text{FVM}} = 4.60$. This is clearly suboptimal as the within-class distance is higher then the between-class distance (to counteract this issue, we have proposed JEANIE).

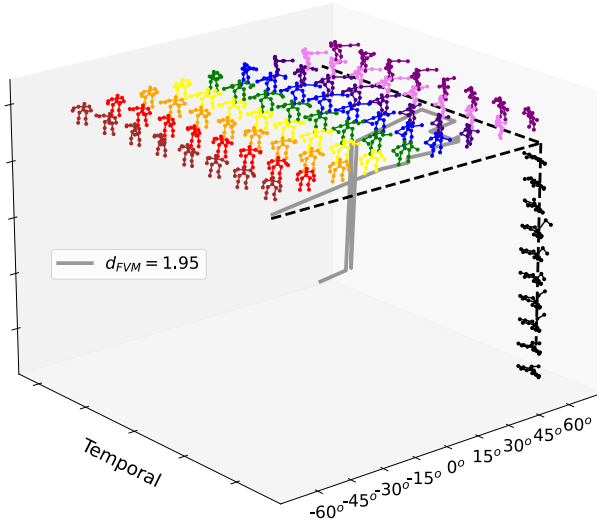


(a) *walking vs. walking* ($d_{\text{JEANIE}} = 8.57$)

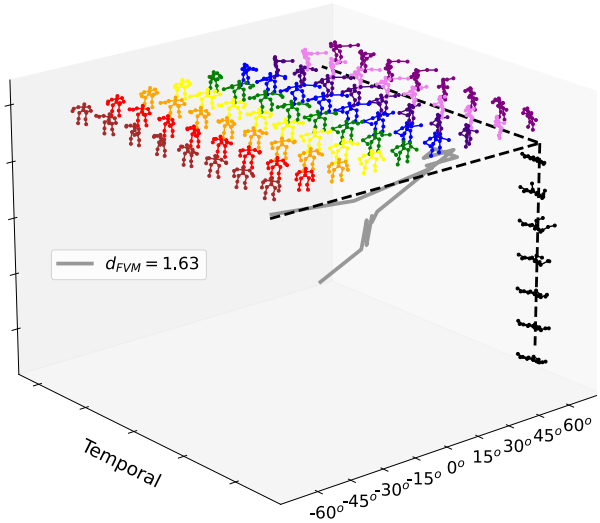


(b) *walking vs. running* ($d_{\text{JEANIE}} = 11.21$)

Fig. 11: Visualization of JEANIE for *walking vs. walking* (two different sequences) and *walking vs. running*. From UWA3D Multiview Activity II, we choose a *walking* sequence as the query sample ('`a12_s01_e01_v01`'). We also choose another *walking* sequence from a different view ('`a12_s01_e01_v03`') and a *running* sequence ('`a20_s01_e01_v02`') as the support samples respectively. In contrast to FVM in Fig. 10, our JEANIE is able to produce a smaller distance for within-class sequences and a larger distance for between-class sequences, which is a very important property when comparing pairs of sequences.

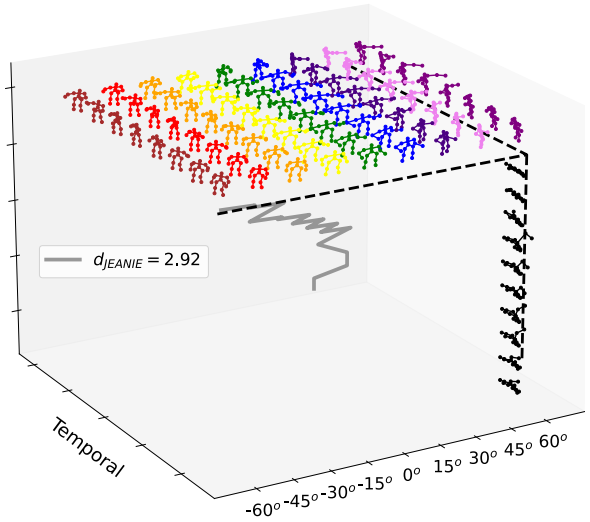


(a) *two hand punching vs. two hand punching* ($d_{\text{FVM}} = 1.95$)

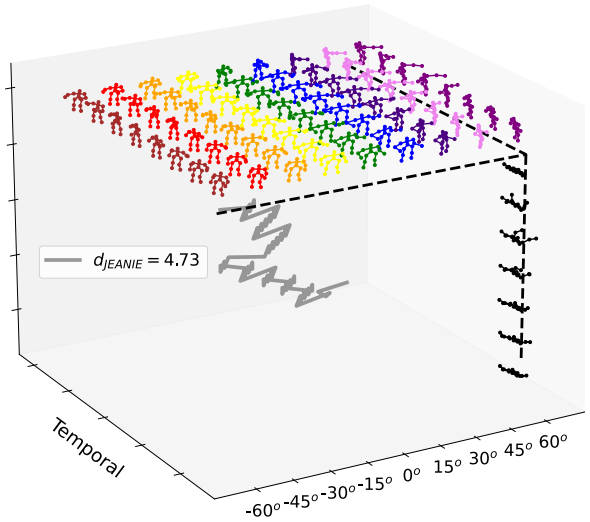


(b) *two hand punching vs. holding head* ($d_{\text{FVM}} = 1.63$)

Fig. 12: Visualization of FVM for *two hand punching vs. two hand punching* (two different sequences) and *two hand punching vs. holding head*. From UWA3D Multiview Activity II, we choose a *two hand punching* sequence as the query sample (‘`a04_s01_e01_v01`’), and another *two hand punching* sequence from a different view (‘`a04_s05_e01_v02`’) and a *holding head* sequence (‘`a10_s05_e01_v02`’) as the support samples respectively. We notice that for two different action sequences in (b), the greedy FVM finds the path which results in $d_{\text{FVM}} = 1.63$ for sequences of different action classes, yet FVM gives $d_{\text{FVM}} = 1.95$ for two sequences of the same class. The within-class distance should be smaller than the between-class distance but greedy approaches such as FVM cannot handle this requirement well.



(a) *two hand punching vs. two hand punching* ($d_{\text{JEANIE}} = 2.92$)



(b) *two hand punching vs. holding head* ($d_{\text{JEANIE}} = 4.73$)

Fig. 13: Visualization of JEANIE for *two hand punching vs. two hand punching* (two different sequences) and *two hand punching vs. holding head*. From UWA3D Multiview Activity II, we choose a *two hand punching* sequence as the query sample (‘`a04_s01_e01_v01`’), and another *two hand punching* sequence from a different view (‘`a04_s05_e01_v02`’) and a *holding head* sequence (‘`a10_s05_e01_v02`’) as the support samples respectively. Our JEANIE gives smaller distance when comparing within-class sequences compared to between-class sequences. This is a very important property when comparing pairs of sequences.

105. Tas, Y., Koniusz, P.: CNN-based action recognition and supervised domain adaptation on 3d body skeletons via kernel feature maps. In: BMVC (2018)
106. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) NeurIPS. pp. 3630–3638 (2016)
107. Wu, F., Zhang, T., de Souza Jr., A.H., Fifty, C., Yu, T., Weinberger, K.Q.: Simplifying graph convolutional networks. In: ICML (2019)
108. Yan, S., Xiong, Y., Lin, D.: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In: AAAI (2018)
109. Zhu, H., Koniusz, P.: Simple spectral graph convolution. In: ICLR (2021)