3Mformer: Multi-order Multi-mode Transformer for Skeletal Action Recognition – Supplementary Material –

Lei Wang[†,§] Piotr Koniusz[*,§,†]
[†]Australian National University, [§]Data61♥CSIRO
[§]firstname.lastname@data61.csiro.au

A. Visualization of 3Mformer

Fig. 4 shows the visualization of our 3Mformer. The green and orange blocks denote the Multi-order Pooling (MP) and the Temporal block Pooling (TP) respectively, which are two basic building blocks that can be stacked to form our 3Mformer. More precisely, our 3Mformer consists of two branches: (i) MP followed by TP (denoted MP → TP, Fig. 4a) and (ii) TP followed by MP (denoted TP → MP, Fig. 4b).

B. Skeletal Graph and Hypergraph

Skeletal Graph [64]. Let $G=(V,E)$ be a skeletal graph with the vertex set V of nodes (body joints) $\{v_1,...,v_J\}$, and E be edges (bones) of the graph, and E consists of E_S and E_T. The subset $E_S=\{(v_{it},v_{jt}) : i,j\in\mathcal{I}_J \text{ and } t\in\mathcal{I}_T\}$ represents that at time step t, each pair of joints (v_{it},v_{jt}) corresponding to skeletal connectivity diagram is connected; whereas $E_T=\{(v_{it},v_{i(t+1)}) : i\in\mathcal{I}_J \text{ and } t\in\mathcal{I}_T\}$ forms the connection of the same joint across time. The set of joints and edges together form the skeleton graph. If two body joints are connected by an edge, the corresponding element in the incidence matrix $\mathbf{H}$ is equal to 1, otherwise it is equal to 0, and the adjacency matrix $\mathbf{A}=\mathbf{H}^T\mathbf{H}-2\mathbf{I}$ (where $\mathbf{I}$ is the identity matrix). The update rule of a common GCN model at time step t is defined as:

$$\mathbf{X}_t^{(l+1)}=\sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}_t^{(l)}\mathbf{\Theta}^{(l)}), \tag{11}$$

where $\sigma(\cdot)$ is a non-linearity, $\widetilde{\mathbf{D}}$ is the graph degree matrix, $\mathbf{X}_t^{(l)}$ is the input data of the convolutional layer l at the time step t and $\mathbf{\Theta}^{(l)}$ is the learnable parameters of layer l. $\widetilde{\mathbf{A}}=\mathbf{A}+\mathbf{I}$ is a normalized graph adjacency matrix.

The tensor representation of graph data can be given by $\mathbf{X}\in\mathbb{R}^{J\times d}$ where d is the feature channel dimension.

Skeletal Hypergraph [15, 35]. Hypergraph captures complex higher-order relationships by hyper-edges that connect more than two nodes (body joints). Each hyper-edge is a subset of all nodes. Let $G_h=(V_h,E_h,W_h)$ where V_h, E_h and W_h denote respectively the set of body joints, hyper-edges and the weights of hyper-edges. Given $v\in V_h$ and $e\in E_h$, the elements in the incidence matrix $\mathbf{H}_h$ of the skeleton hypergraph are defined as $H_{h,v,e}=1$, or simply put $h(v,e)=1$, if vertex v is part of edge e, 0 otherwise. The degree of node/body joint $v\in V_h$ is the number of hyper-edges passing through the node, which is defined as:

$$d(v)=\sum_{e\in E_h} w(e)h(v,e), \tag{12}$$

where $w(e)$ is the weight of hyper-edge e. The degree of hyper-edge $e\in E_h$ is the number of nodes (body joints) contained in the hyper-edge e that satisfies:

$$\delta(e)=\sum_{v\in V_h} h(v,e). \tag{13}$$

Moreover, let $\mathbf{D}_v$ and $\mathbf{D}_e$ be the diagonal matrices of node degrees $d(v)$ and the hyper-edge degrees $\delta(e)$ respectively. Let $\mathbf{W}$ denote the diagonal matrix of the hyper-edge weights (initially the weights of all hyper-edges are set to 1). Then the update rule of the Hypergraph Convolutional Network at the time step t is given by:

$$\mathbf{X}_t^{(l+1)}=\sigma(\mathbf{D}_v^{\frac{1}{2}}\mathbf{H}_h\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}_h^{\top}\mathbf{D}_v^{\frac{1}{2}}\mathbf{X}_t^{(l)}\mathbf{\Theta}^{(l)}), \tag{14}$$

where $\mathbf{\Theta}^{(l)}$ are learnable parameters for layer l.

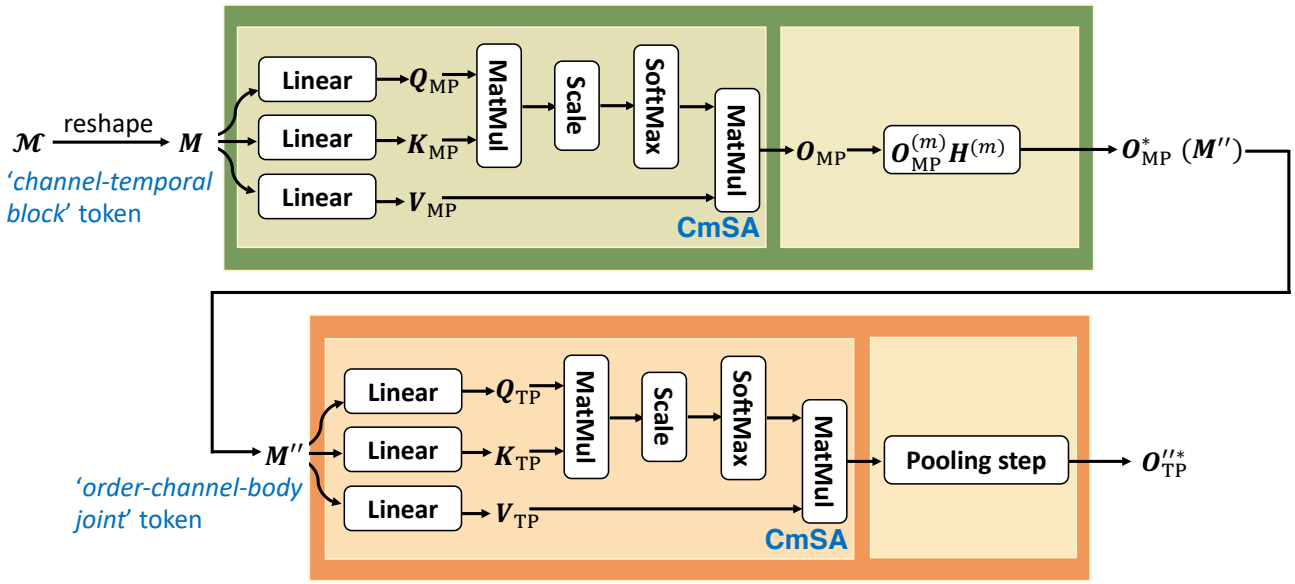
C. Skeleton Data Preprocessing

Before passing the skeleton sequences into MLP, we first normalize each body joint *w.r.t.* to the torso joint $\mathbf{v}_{f,c}$:

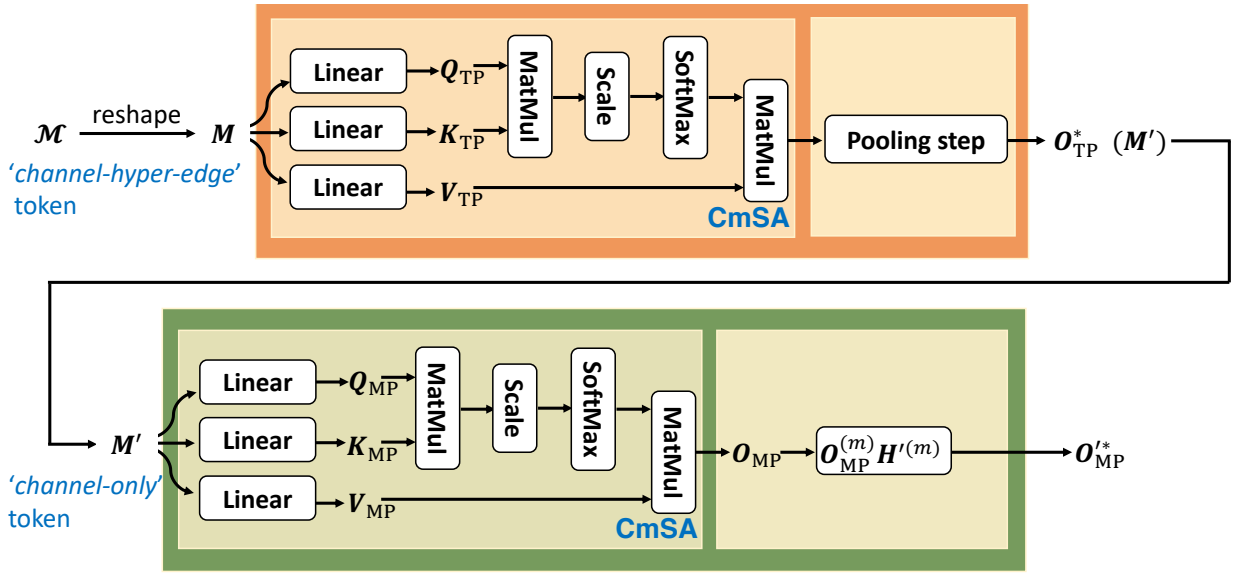
$$\mathbf{v}'_{f,i}=\mathbf{v}_{f,i}-\mathbf{v}_{f,c}, \tag{15}$$

where f and i are the index of video frame and human body joint respectively. After that, we further normalize each

*Corresponding author.



(a) Single-branch: MP followed by TP (denoted MP→TP).



(b) Single-branch: TP followed by MP (denoted TP→MP).

Figure 4. Visualization of 3Mformer which is a two- branch model: (a) MP→TP and (b) TP→MP. Green and orange blocks are Multi-order Pooling (MP) module and Temporal block Pooling (TP) module, respectively. (m) inside the MP module denotes the order $m \in \mathcal{I}_r$ of hyper-edges. These two modules (MP and TP) are the basic building blocks which are further stacked to form our 3Mformer. Each module (MP or TP) uses a specific coupled-mode token through matricization (we use reshape for simplicity), *e.g.*, 'channel-temporal block', 'order-channel-body joint', 'channel-hyper-edge (any order)' or 'channel-only', and the Coupled-mode Self-Attention (CmSA) is used to explore the coupled-mode relationships inside the coupled-mode tokens. We also form our multi-head CmSA as in standard Transformer (where the CmSA module repeats its computations multiple times in parallel and the attention module splits the query, key and value, each split is independently passed through a separate head and later combined together to produce the final coupled-mode attention score). We omit the multi-head visualization for simplicity and better visualization purposes.

joint coordinate into [-1, 1] range:

$$\hat{\mathbf{v}}_{f,i}[j] = \frac{\mathbf{v}'_{f,i}[j]}{\max([\text{abs}(\mathbf{v}'_{f,i}[j])]_{f\in\mathcal{I}_\tau, i\in\mathcal{I}_J})}, \tag{16}$$

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,

Table 5. Ablations of different pooling methods in MP.

Pooling	NTU-60		NTU-120		Kinetics-Skel.
	X-Sub	X-View	X-Sub	X-Set	Top-1 acc.
avg-pool	91.3	96.8	86.5	89.0	41.9
max-pool	92.7	98.0	88.5	91.0	43.8
wei-pool (*ours*)	94.8	98.7	92.0	93.8	48.3

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 the block-temporal HoT, *e.g.*, two skeletons from a given video sequence will have two outputs obtained from the the block-temporal HoT, and we aggregate the outputs through average pooling before passing our 3Mformer.

D. Additional Results and Discussions

D.1. Ablations of MP

We choose average pooling (avg-pool) and max-pooling (max-pool) for hyper-edge features in comparison to our learned weighted pooling (wei-pool), and the comparisons are given in Table 5. As shown in the table, our learned weighted pooling (wei-pool) consistently achieves the best performance on all 3 datasets.

D.2. Learning the short-term temporal patterns

A block of T neighbor frames are passed to the MLP unit to capture the short-term temporal patterns. The whole sequence consists of τ such blocks, each passed separately through the MLP unit (and each joint $1, ..., J$). Thus, the MLP only mixes the information from $1, ..., T$ frames of a given block/body joint j and captures short-term relations (within-block) of a given 3D body joint (in contrast to between-block relations). The MLP unit input size is $3T$; 3 due to 3D coordinate). The MLP : $\mathbb{R}^{3T} \rightarrow \mathbb{R}^{d}$ contains: FC ($3T \rightarrow 6T$), ReLU, FC ($6T \rightarrow 9T$), ReLU, Dropout, FC ($9T \rightarrow d$). J body joints and τ blocks are treated as the batch dimension. Feature output size d: 100, 150, 420 on NTU-60, NTU-120, Kinetics-Skeleton.

D.3. Why 3Mformer works and when does it fail?

Our method works well as it (i) uses skeletal hypergraphs of various orders to learn the interaction of varying size groups of skeletal joints (as opposed to typical skeleton graph physical connectivity), (ii) fuses groups multiple orders by 3Mformer by several coupled-token types via two basic building blocks (MP & TP) that learn various aspects of higher-order motion dynamics. Multiple-order hyperedges are more resistant to noise (*e.g.*, Kinetics-Skeleton is noisy due to the pose estimation errors), if one body joint is noisy (but the rest is stable). We inject Gaussian noise into 3D *ankle* joints, vary noise amplitude, and we show the experimental results in Table 6. As shown in the table, our 3Mformer copes with noise better than ST-GCN.

Table 6. Comparisons of robustness *w.r.t.* Gaussian noise.

	original	$\times$ 1	$\times$ 1.5	$\times$ 2
ST-GCN	81.5	74.9 (↓6.6)	69.2 (↓12.3)	50.1 (↓31.4)
3Mformer	94.8	91.9 (↓2.9)	89.5 (↓5.3)	86.8 (↓8.0)

Table 7. Experimental results on MSRAction3D.

order	2	3	4
acc.(%)	73.82	63.64	55.27

Table 8. A comparison of the number of model parameters and FLOPs on NTU-60.

	ST-GCN	2S-AGCN	NAS-GCN	2rd-order only (*ours*)	3rd-order only (*ours*)	3Mformer (*ours*)
Params (M)	3.14	3.45	6.57	1.15	2.07	4.37
FLOPs (G)	16.36	37.22	108.82	6.54	35.53	58.45
Acc. (%)	81.5	88.5	89.4	83.0	91.3	**94.8**

Our method may underperform if (i) the backbone encoder cannot efficiently produce higher-order features (ii) the number of skeletal joints are very large (the number of hyper-edge features would be very large) (iii) when dataset is too small to learn high-order interactions (extra learnable parameters). For example, see the experimental results on MSRAction3D in Table 7.

We notice that small datasets may be not enough to train high-order models (Table 7). On key classic large datasets, NTU-60, NTU-120 and Kinetics-Skeleton, we do not observe any issue as human motions exhibit similar multi-joint dynamics for typical action classes. Perhaps some fine-grained unusual action classes could pose problems.

D.4. Model Complexity

Table 8 shows the number of model parameters/FLOPs and NTU-60 accuracy. Our cost is moderate. 2S-AGCN (37.22 GFLOPs & 3.45M param.) yields 89.4% accuracy. Our '3rd-order' uses 35.5 GFLOPs & 2.07M param. which is 2 GFLOPs & 1.37M param. less, yet we outperform 2S-AGC by **1.9%**. NAS-GCN uses 40.4 GFLOPs/2.2M param. more compared to our 3Mformer: we beat NAS-GCN by **4.4%**.

D.5. Limitation and Future Work

Despite the high accuracy of our model, there are still some limitations. Firstly, as we use r branches of HoT, the number of parameters and computational cost are higher than existing methods. However, our method with single branch, *e.g.*, 3rd-order HoT only, still achieves very competitive results compared to existing graph-, transformer- and hypergraph-based models for the same parameter scale on 3 benchmarks. Secondly, in this work, we only use HoT block to encode the temporal block feature representations. The

more efficient way is to redesign HoT block so that it is able to encode both short-term and long-term spatio-temporal features to simplify the backbone encoder, *i.e.*, without the need of MLP unit. Note that the design of our 3Mformer is independent of the backbone encoder. Our 3Mformer is especially suitable for tensorial data, *e.g.*, higher-order feature representations. Our future work will focus on applying our Multi-order Multi-mode Transformer (3Mformer) to other computer vision tasks with tensorial data.