Supplementary Material

Supple. Sec.	Descriptions for analysis and discussion
Sec. A	**High-level novelty compared to conventional methods**
Sec. B.1	**Statistical analysis of Skate-MSA for different classes** (i) Accuracy comparisons of SkateFormer for different classes (Table. 8) (ii) Statistical analysis of action classes where Skate-MSA enhances performance (iii) Discussion of failure action classes
Sec. B.2	**Generalization issues to diverse modality inputs**
Sec. B.3	**Visualizing the activation levels of Skate-MSA** (i) Visualization of activation for different classes (Fig. 5) (ii) Discussion of adaptive nature of Skate-MSA concerning action classes
Sec. E.1	**Theoretical calculation of the computational complexity of Skate-MSA**
Sec. E.3	**Additional experiments on the computational complexity**

Due to the space limit of the main paper, we had a difficulty to contain all the detailed descriptions for the analysis and discussion on the various experimental results in the same granularity levels of details. Instead, high level descriptions are included in the main paper while detailed descriptions are in the *Supple.* as shown in the above table.

A Discussion on Various Partition-Based Approaches

In recent studies, several partition-based methods such as DSTA-Net [44], ST-TR [38], IIP-Trnasformer [60], STST [73], FG-STFormer [13], Hyperformer [76], IGFormer [34], SkeleTR [11], and ISTA-Net [62] have been proposed. As shown in Table 7, we introduce SkateFormer, which differs fundamentally in the following aspects:

Table 7: Comparison with ours SkateFormer with existing transformer-based methods. S-Attn, T-Attn, T-Conv, and ST-Attn indicate 'skeletal attention', 'temporal attention', 'temporal convolution' and 'skeletal-temporal attention', respectively.

Tasks	Methods	Partition Types	Tokenization for Attention	Attention Types
Human Action Recognition	DSTA-Net [44]	N/A (Reshaping)	No	S-Attn, T-Attn (Sequential)
	ST-TR [38]	N/A (Reshaping)	No	S-Attn, T-Attn (Parallel)
	IIP-Transformer [60]	S-Type	Yes	S-Attn, T-Attn (Sequential)
	STST [73]	N/A (Reshaping)	No	S-Attn, T-Attn (Parallel)
	FG-STFormer [13]	S-Type	Yes	S-Attn, T-Attn (Sequential)
	Hyperformer [76]	S-Type	Yes	S-Attn, T-Conv (Sequential)
Human Interaction Recognition	IGFormer [34]	S-Type, T-Type	Yes	Joint-group-level ST-Attn
	SkeleTR [11]	N/A (Pooling)	Yes	Joint-group-level ST-Attn
	ISTA-Net [62]	S-Type, T-Type	Yes	Joint-group-level ST-Attn
Both	**SkateFormer**	4 Skate-Types	No	Joint-element-level ST-Attn

- Unlike existing methods that rely solely on physically neighboring joints (S-Type) or local motion (T-Type), our SkateFormer leverages four skeletal-temporal relation types (Skate-Types) with joint partitions (physically 'neighboring' + 'distant') and frame partitions ('local' + 'global' motions).
- While prior approaches typically tokenize joints within the same partition into a single feature (partition-based tokenization, not partition-specific attention), our SkateFormer adopts self-attention within each partition, enhancing its capability for fine-grained analysis.
- To mitigate computational complexity, existing methods often employ strategies such as conducting attention solely in the skeletal dimension while utilizing temporal convolution in the temporal dimension, or separately performing skeletal and temporal attention mechanisms, or employing joint-group-level attention with tokenization. In contrast, our SkateFormer is specifically designed to efficiently capture skeletal-temporal relations at the joint-element-level, thus eliminating the need for tokenization.

B Additional Results

B.1 Recognition Accuracy based on Action Labels

Table 8 presents the top-1 accuracy results for single-joint modality (J) action recognition in the NTU RGB+D X-Sub60 evaluation, categorized by action labels. The baseline model represents a model without the application of partition-specific attention based on four skeletal-temporal relation types from Skate-MSA (no attention, only $\mathbf{V}_h$ in Eq. 8). The model denoted as ($+ \mathbf{v}_k^{\mathsf{njp}} + \mathbf{v}_l^{\mathsf{djp}}$) applies partition-specific attention based on two skeletal relation types, while ($+ \mathbf{t}_m^{\mathsf{local}} + \mathbf{t}_n^{\mathsf{global}}$) utilizes partition-specific attention based on two temporal relation types. The model labeled ($+\mathbf{v}_k^{\mathsf{njp}} + \mathbf{v}_l^{\mathsf{djp}} + \mathbf{t}_m^{\mathsf{local}} + \mathbf{t}_n^{\mathsf{global}}$) represents the full SkateFormer model, which incorporates partition-specific attention based on four Skate-Types.

Our baseline model already outperforms state-of-the-art methods [3,7,10,22, 63,67,75], as evident from the average top-1 accuracy. The *Rank* in the baseline indicates the ranking of top-1 accuracy among 60 classes. Comparing the full model to the baseline, we observed performance improvements in the majority of action labels (42 out of 60 classes), with eight classes maintaining the same performance, and a slight decrease in performance for ten classes. This can be interpreted as an effective utilization of limited model capacity, where slight decreases in performance for high-performing action labels (high-rank) allow for significant improvements in performance for lower-performing action labels (low-rank).

Notably, the baseline performance for actions such as 'wear a shoe' and 'take off a shoe' increased substantially from (60.81%, 78.83%) to (87.55%, 83.94%), demonstrating that our partition-specific attention enhances discriminative capability for similar classes. Furthermore, performance improvements for relatively low-rank action labels, such as 'typing on a keyboard', 'sneeze/cough', 'use a fan (with hand or paper)/feeling warm', were substantial.

Analysis of failure cases. As shown in Table 8, most of the failure cases occur when action classes are determined by fine finger motions, e.g. 'reading', 'writing', 'playing with phone/tablet', and 'typing on a keyboard'.

B.2 Single Modality Comparision

In Tables 9, 10, we present the results of action recognition based on single modalities. J denotes the joint modality, B represents the bone modality, JM indicates joint motion modality, and BM signifies bone motion modality. The top-1 accuracy is reported based on both the published paper and the official code provided. As an exception, for studies marked with (*), we relied on the performance reported in [10], as the official code and paper did not provide modality-specific performance. Table 9 shows the top-1 accuracy for X-Sub60 and X-View60 evaluation on the NTU RGB+D [41] dataset. Table 10 displays the top-1 accuracy for X-Sub120 and X-Set120 evaluation on the NTU RGB+D 120 [27] dataset.

In the domain of skeleton-based action recognition, modality ensemble is a common approach where networks with the same architecture are trained separately for different modalities, and the final label is determined through a weighted summation of network outcomes [3, 7, 75, 76]. As a consequence, the computational requirements, including FLOPs, parameter count, and inference time, increase in proportion to the number of modalities being ensembled. Therefore, achieving effective recognition performance with just a single modality is of paramount importance for real-world applications. As evident from Tables 9, 10, our results showcase a significant dominance in single-modality performance over existing state-of-the-art methods, particularly in the J and B modalities. Furthermore, our findings demonstrate that a single modality alone can achieve performance comparable to that of ensemble modalities in previous methods.

B.3 Analysis of Partition-specific Attention

To analyze the significance of different Skate-Types concerning action labels, we conducted an investigation within Skate-MSA. We assessed the strength of the feature correlation maps $\mathbf{QK}^{\mathsf{T}}$ (prior to $\mathsf{SoftMax}$ in SA, Eq. 8) corresponding to each Skate-Type by calculating their mean values (referred to as *Skate-Type Importance Score*) with respect to action labels. In Fig. 5, we illustrate the Skate-Type Importance Score for several action labels. For actions such as 'sitting down' (action label: 7) and 'standing up (from a sitting position)' (action label: 8), where the overall motion of the skeleton sequence is pivotal, the feature correlation maps for Skate-Type-3 or Skate-Type-4 exhibited more pronounced activations compared to other Skate-Types. Conversely, for actions that rely heavily on intricate hand movements like 'reading' (action label: 10) and 'writing' (action label: 11), the feature correlation map for Skate-Type-1 was prominently activated. Through this analysis, we affirm that our proposed partition-specific attention strategy based on Skate-Types operates adaptively according to action labels.

Table 8: Top-1 accuracy based on action labels in NTU RGB+D X-Sub60 evaluation.

Action Label	Baseline (Rank)	$+\mathbf{v}_k^{\text{njp}} + \mathbf{v}_l^{\text{djp}}$	$+ \mathbf{t}_{rn}^{\text{local}} + \mathbf{t}_n^{\text{global}}$	$+\mathbf{v}_k^{\text{njp}} + \mathbf{v}_l^{\text{djp}} + \mathbf{t}_{rn}^{\text{local}} + \mathbf{t}_n^{\text{global}}$
drink water	85.77 (47)	85.77 (–0.00)	85.77 (–0.00)	88.32 (↑2.55)
eat meal/snack	76.00 (56)	77.82 (↑1.82)	77.45 (↑1.45)	78.18 (↑2.18)
brushing teeth	90.11 (42)	85.35 (↓4.76)	91.21 (↑1.10)	89.01 (↓1.10)
brushing hair	89.01 (45)	91.21 (↑2.20)	93.77 (↑4.76)	93.41 (↑4.40)
drop	92.00 (39)	94.18 (↑2.18)	92.36 (↑0.36)	92.73 (↑0.73)
pickup	97.09 (18)	97.82 (↑0.73)	97.09 (–0.00)	98.55 (↑1.45)
throw	92.73 (36)	93.45 (↑0.73)	92.00 (↓0.73)	93.82 (↑1.09)
sitting down	98.17 (13)	98.53 (↑0.37)	99.27 (↑1.10)	98.53 (↑0.37)
standing up (from sitting position)	98.90 (7)	98.53 (↓0.37)	99.63 (↑0.73)	98.90 (–0.00)
clapping	83.88 (51)	81.68 (↓2.20)	86.08 (↑2.20)	85.35 (↑1.47)
reading	58.61 (60)	60.81 (↑2.20)	61.54 (↑2.93)	63.37 (↑4.76)
writing	68.38 (57)	68.01 (↓0.37)	66.91 (↓1.47)	71.32 (↑2.94)
tear up paper	95.94 (21)	96.68 (↑0.74)	94.46 (↓1.48)	95.94 (–0.00)
wear jacket	98.18 (12)	98.18 (–0.00)	98.55 (↑0.36)	97.82 (↓0.36)
take off jacket	98.19 (10)	98.91 (↑0.72)	99.64 (↑1.45)	99.64 (↑1.45)
wear a shoe	60.81 (59)	86.81 (↑26.01)	84.62 (↑23.81)	87.55 (↑**26.74**)
take off a shoe	78.83 (53)	82.48 (↑3.65)	78.47 (↑0.36)	83.94 (↑**5.11**)
wear on glasses	93.04 (34)	93.41 (↑0.37)	93.77 (↑0.73)	93.41 (↑0.37)
take off glasses	95.62 (23)	95.62 (–0.00)	95.26 (↓0.36)	94.89 (↓0.73)
put on a hat/cap	96.69 (19)	97.43 (↑0.74)	98.16 (↑1.47)	97.79 (↑1.10)
take off a hat/cap	98.90 (7)	98.53 (↓0.37)	98.90 (–0.00)	98.90 (–0.00)
cheer up	93.80 (29)	92.34 (↓1.46)	93.80 (–0.00)	94.89 (↑1.09)
hand waving	93.80 (29)	94.16 (↑0.36)	93.80 (–0.00)	93.07 (↓0.73)
kicking something	94.20 (27)	94.93 (↑0.72)	97.10 (↑2.90)	97.83 (↑3.62)
reach into pocket	84.67 (49)	84.31 (↓0.36)	86.13 (↑1.46)	86.86 (↑2.19)
hopping (one foot jumping)	98.91 (6)	98.91 (–0.00)	98.91 (–0.00)	98.91 (–0.00)
jump up	100.00 (1)	100.00 (–0.00)	100.00 (–0.00)	100.00 (–0.00)
make a phone call/answer phone	90.18 (41)	89.09 (↓1.09)	90.55 (↑0.36)	92.00 (↑1.82)
playing with phone/tablet	76.36 (55)	76.36 (–0.00)	77.09 (↑0.73)	74.91 (↓1.45)
typing on a keyboard	64.36 (58)	72.73 (↑8.36)	74.55 (↑10.18)	77.82 (↑**13.45**)
pointing to something with finger	79.71 (52)	88.04 (↑8.33)	81.88 (↑2.17)	84.42 (↑4.71)
taking a selfie	92.39 (38)	93.84 (↑1.45)	94.57 (↑2.17)	93.12 (↑0.72)
check time (from watch)	93.12 (33)	91.67 (↓1.45)	92.75 (↓0.36)	92.03 (↓1.09)
rub two hands together	89.13 (43)	89.49 (↑0.36)	91.30 (↑2.17)	92.03 (↑2.90)
nod head/bow	97.46 (14)	98.55 (↑1.09)	98.55 (↑1.09)	99.28 (↑1.81)
shake head	94.55 (26)	95.27 (↑0.73)	96.73 (↑2.18)	96.00 (↑1.45)
wipe face	85.87 (46)	87.32 (↑1.45)	87.68 (↑1.81)	90.94 (↑5.07)
salute	93.48 (31)	93.48 (–0.00)	93.84 (↑0.36)	94.93 (↑1.45)
put the palms together	97.46 (14)	97.46 (–0.00)	96.74 (↓0.72)	96.38 (↓1.09)
cross hands in front (say stop)	96.01 (20)	97.10 (↑1.09)	96.38 (↑0.36)	97.46 (↑1.45)
sneeze/cough	78.62 (54)	82.97 (↑4.35)	83.33 (↑4.71)	85.51 (↑**6.88**)
staggering	99.28 (4)	99.64 (↑0.36)	100.00 (↑0.72)	99.28 (–0.00)
falling	99.64 (2)	100.00 (↑0.36)	99.64 (–0.00)	100.00 (↑0.36)
touch head (headache)	84.06 (50)	89.13 (↑5.07)	84.06 (–0.00)	88.41 (↑4.35)
touch chest (stomachache/heart pain)	93.84 (28)	94.57 (↑0.72)	93.48 (↓0.36)	96.01 (↑2.17)
touch back (backache)	95.29 (24)	94.57 (↓0.72)	95.29 (–0.00)	95.65 (↑0.36)
touch neck (neckache)	90.22 (40)	90.58 (↑0.36)	92.39 (↑2.17)	93.12 (↑2.90)
nausea or vomiting condition	85.45 (48)	83.64 (↓1.82)	84.00 (↓1.45)	85.45 (–0.00)
use a fan (with hand or paper)/feeling warm	89.09 (44)	90.91 (↑1.82)	89.45 (↑0.36)	94.18 (↑**5.09**)
punching/slapping other person	92.70 (37)	93.07 (↑0.36)	92.70 (–0.00)	94.16 (↑1.46)
kicking other person	95.65 (22)	96.38 (↑0.72)	95.65 (–0.00)	96.38 (↑0.72)
pushing other person	98.55 (9)	97.83 (↓0.72)	97.46 (↓1.09)	98.19 (↓0.36)
pat on back of other person	93.48 (31)	95.29 (↑1.81)	94.93 (↑1.45)	95.65 (↑2.17)
point finger at the other person	92.75 (35)	94.20 (↑1.45)	94.57 (↑1.81)	94.20 (↑1.45)
hugging other person	99.27 (5)	99.27 (–0.00)	99.64 (↑0.36)	99.64 (↑0.36)
giving something to other person	95.29 (24)	96.74 (↑1.45)	95.65 (↑0.36)	95.65 (↑0.36)
touch other person's pocket	97.45 (17)	97.45 (–0.00)	98.55 (↑1.09)	96.00 (↓1.45)
handshaking	97.46 (14)	97.10 (↓0.36)	97.46 (–0.00)	97.46 (–0.00)
walking towards each other	99.63 (3)	100.00 (↑0.37)	100.00 (↑0.37)	100.00 (↑0.37)
walking apart from each other	98.19 (10)	97.10 (↓1.09)	96.74 (↓1.45)	97.83 (↓0.36)
average	90.65	91.79 (↑1.14)	91.88 (↑1.23)	92.62 (↑1.98)

Table 9: Top-1 classification accuracy of different skeleton-based action recognition methods on NTU RGB+D [41] dataset. The **best performances** are highlighted in **bold**.

Methods	Frames	NTU RGB+D (%)							
		X-Sub60				X-View60			
		J	B	JM	BM	J	B	JM	BM
ST-GCN (*) [67]	100	87.8	88.6	85.8	86.2	95.5	95.0	93.7	92.8
CTR-GCN [3]	64	89.9	90.6	88.1	87.9	-	-	-	-
CTR-GCN (*) [3]	100	89.6	90.0	88.0	87.5	95.6	95.4	94.4	93.6
ST-GCN++ [10]	100	89.3	90.1	87.5	87.3	95.6	95.5	94.3	93.8
InfoGCN [7]	64	89.8	90.6	88.9	88.6	95.2	95.5	94.2	93.6
FR-Head [75]	64	90.3	91.1	88.7	87.6	95.3	95.0	93.6	92.6
LST [63]	64	90.2	91.2	88.0	87.8	95.6	95.5	93.7	93.2
HD-GCN [22]	64	90.6	90.9	-	-	95.7	95.1	-	-
SkateFormer	64	**92.6**	**92.1**	**89.8**	**89.0**	**97.0**	**96.5**	**95.8**	**94.7**

Table 10: Top-1 classification accuracy of different skeleton-based action recognition methods on NTU RGB+D 120 [27] dataset. The **best performances** are highlighted in **bold**.

Methods	Frames	NTU RGB+D 120 (%)							
		X-Sub120				X-Set120			
		J	B	JM	BM	J	B	JM	BM
ST-GCN (*) [67]	100	82.1	83.7	80.3	80.6	84.5	85.8	82.7	83.0
CTR-GCN [3]	64	84.9	85.7	81.4	81.2	-	87.5	-	-
CTR-GCN (*) [3]	100	84.0	85.9	81.1	82.2	85.9	87.4	84.1	83.9
ST-GCN++ [10]	100	83.2	85.6	80.4	81.5	85.6	87.5	84.3	83.0
InfoGCN [7]	64	85.1	87.3	82.1	**82.5**	86.3	88.5	84.4	**84.8**
FR-Head [75]	64	85.5	86.8	81.9	82.0	87.3	88.1	84.0	83.9
LST [63]	64	85.5	87.5	82.3	82.4	87.0	88.7	83.9	84.4
HD-GCN [22]	64	85.7	86.7	-	-	87.3	88.4	-	-
SkateFormer	64	**87.7**	**88.2**	**83.1**	82.3	**89.3**	**89.8**	**85.3**	84.1

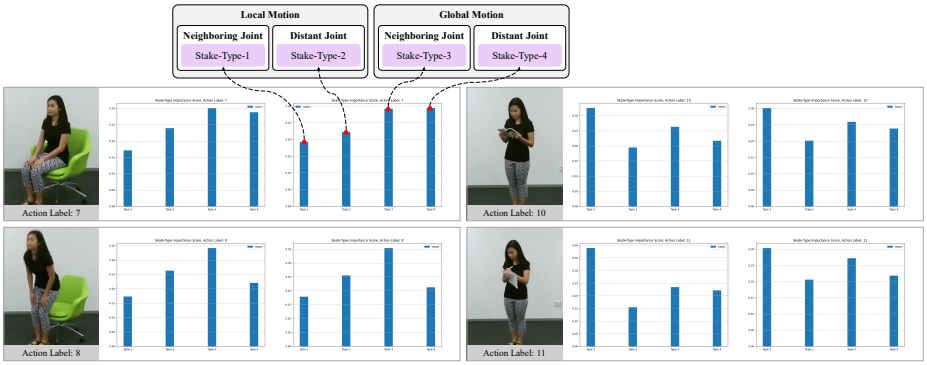

Fig. 5: Skate-Type Importance Scores for various action labels.

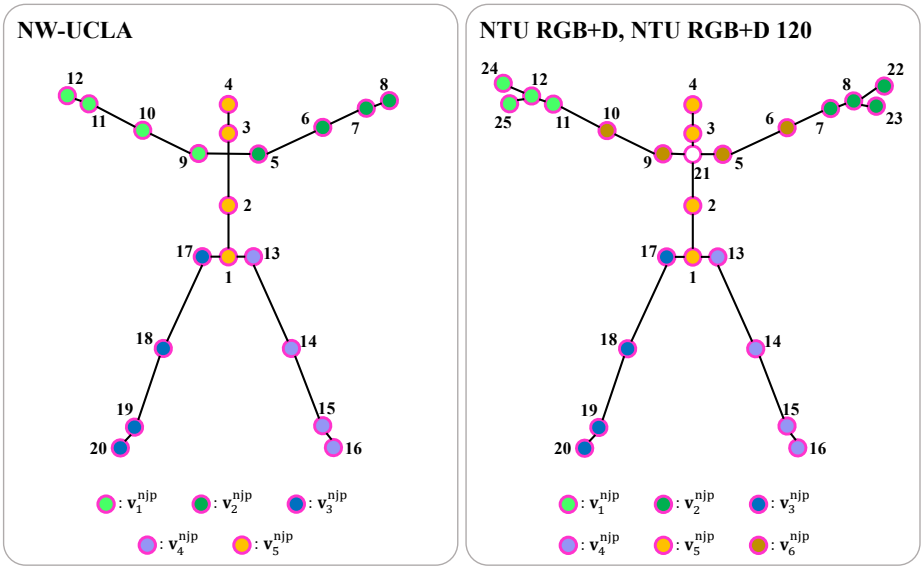

Fig. 6: Skeleton tracking indices and neighboring joint partitions.

C Joint Details for Various Datasets

C.1 Skeleton Tracking Indices and Labels of Joints

NW-UCLA. In Fig. 6, for the NW-UCLA [56] dataset acquired through the Kinect v1 [16] sensor, the skeleton of a single individual consists of a total of 20 joints. Each joint, which is a constituent unit of the skeleton, can be represented by default indices from 1 to 20, as shown in Fig. 6. The labels for each index are as follows [27]: (1) base of spine, (2) middle of spine, (3) neck, (4) head, (5) left shoulder, (6) left elbow, (7) left wrist, (8) left hand, (9) right shoulder, (10) right elbow, (11) right wrist, (12) right hand, (13) left hip, (14) left knee, (15) left ankle, (16) left foot, (17) right hip, (18) right knee, (19) right ankle, (20) right foot. We will denote joints according to their tracking indices k as v_k^{tra}.

NTU RGB+D and NTU RGB+D 120. In Fig. 6, for the NTU RGB+D [41] and NTU RGB+D 120 [27] datasets acquired through the Kinect v2 [16] sensor, the skeleton of a single individual consists of a total of 25 joints. These 25 joints include the original 20 joints from the Kinect v1 sensor, along with 5 additional joints representing the tips of both hands and the thumbs, as well as a spine joint. Similar to the NW-UCLA dataset, each joint can be represented by default indices from 1 to 25, as shown in Fig. 6. The labels for each index are as follows [27]: (1) base of spine, (2) middle of spine, (3) neck, (4) head, (5) left shoulder, (6) left elbow, (7) left wrist, (8) left hand, (9) right shoulder, (10) right elbow, (11) right wrist, (12) right hand, (13) left hip, (14) left knee, (15) left ankle, (16) left foot, (17) right hip, (18) right knee, (19) right ankle, (20) right

foot, (21) spine, (22) tip of left hand, (23) left thumb, (24) tip of right hand, (25) right thumb.

C.2 Details of Neighboring Joint Partitions

In this paper, we introduce the concept of neighboring joint partitions ($\mathbf{v}_k^{\mathsf{njp}}$). We partitioned the entire set of joints into a total of K non-overlapping subsets for our analysis. We empirically set K proportional to the number of joints. We used the same number of joints for each partition since the implementation complexity was surged in usage of 'reshaping' and 'attention' functions due to the irregularity of input size. Alternative solutions as future work might be (i) reusing the same joints, (ii) trimming similar joints, or (iii) resizing the partition by linear interpolation between joints. The ordering of joints in the torso is not critical since their relative positions remain nearly constant (rigid) for any actions. We conducted experiments by varying the index order of the joints in the torso, which yielded little difference in performance. Below, we provide detailed information about the neighboring joint partitions for each dataset.

NW-UCLA. For the NW-UCLA dataset, we set $K = 5$. Therefore, the total number of joints, $V = 20$, is divided into $L = 20/5 = 4$ elements, creating neighboring joint partitions. Specifically, $\mathbf{v}_1^{\mathsf{njp}}$ represents the joints of the right arm, $\mathbf{v}_2^{\mathsf{njp}}$ represents the joints of the left arm, $\mathbf{v}_3^{\mathsf{njp}}$ represents the joints of the right leg, $\mathbf{v}_4^{\mathsf{njp}}$ represents the joints of the left leg, and $\mathbf{v}_5^{\mathsf{njp}}$ represents the vertical torso. These partitions are ordered in a manner that extends outward from the body's central region:

$$\begin{aligned}
\mathbf{v}_1^{\mathsf{njp}} &= \left[v_9^{\mathsf{tra}}, v_{10}^{\mathsf{tra}}, v_{11}^{\mathsf{tra}}, v_{12}^{\mathsf{tra}}\right] \\
\mathbf{v}_2^{\mathsf{njp}} &= \left[v_5^{\mathsf{tra}}, v_6^{\mathsf{tra}}, v_7^{\mathsf{tra}}, v_8^{\mathsf{tra}}\right] \\
\mathbf{v}_3^{\mathsf{njp}} &= \left[v_{17}^{\mathsf{tra}}, v_{18}^{\mathsf{tra}}, v_{19}^{\mathsf{tra}}, v_{20}^{\mathsf{tra}}\right] \\
\mathbf{v}_4^{\mathsf{njp}} &= \left[v_{13}^{\mathsf{tra}}, v_{14}^{\mathsf{tra}}, v_{15}^{\mathsf{tra}}, v_{16}^{\mathsf{tra}}\right] \\
\mathbf{v}_5^{\mathsf{njp}} &= \left[v_2^{\mathsf{tra}}, v_3^{\mathsf{tra}}, v_1^{\mathsf{tra}}, v_4^{\mathsf{tra}}\right].
\end{aligned} \tag{10}$$

NTU RGB+D and NTU RGB+D 120. For the NTU RGB+D and NTU RGB+D 120 datasets, we set $K = 12$. Since each frame in the dataset may contain up to two individuals, and to ensure accurate action recognition, we treat each individual's joints separately. This results in a total of 50 joints per frame. In instances where only one individual is present, the remaining 25 joints are zero-padded. To facilitate partitioning, we exclude the 21st joint and consider the remaining 48 joints, resulting in $V = 48$. Thus, we partition each individual's joints into $L = 48/12 = 4$ elements for neighboring joint partitions: $\mathbf{v}_1^{\mathsf{njp}}$ for the right arm, $\mathbf{v}_2^{\mathsf{njp}}$ for the left arm, $\mathbf{v}_3^{\mathsf{njp}}$ for the right leg, $\mathbf{v}_4^{\mathsf{njp}}$ for the left leg, $\mathbf{v}_5^{\mathsf{njp}}$ for the vertical torso, and $\mathbf{v}_6^{\mathsf{njp}}$ for the horizontal torso. These partitions are ordered in a manner that extends outward from the body's central region:

$$
\begin{aligned}
\mathbf{v}_1^{\mathsf{njp}} &= \left[v_{11}^{\mathsf{tra}}, v_{12}^{\mathsf{tra}}, v_{24}^{\mathsf{tra}}, v_{25}^{\mathsf{tra}}\right] \\
\mathbf{v}_2^{\mathsf{njp}} &= \left[v_{7}^{\mathsf{tra}}, v_{8}^{\mathsf{tra}}, v_{22}^{\mathsf{tra}}, v_{23}^{\mathsf{tra}}\right] \\
\mathbf{v}_3^{\mathsf{njp}} &= \left[v_{17}^{\mathsf{tra}}, v_{18}^{\mathsf{tra}}, v_{19}^{\mathsf{tra}}, v_{20}^{\mathsf{tra}}\right] \\
\mathbf{v}_4^{\mathsf{njp}} &= \left[v_{13}^{\mathsf{tra}}, v_{14}^{\mathsf{tra}}, v_{15}^{\mathsf{tra}}, v_{16}^{\mathsf{tra}}\right] \\
\mathbf{v}_5^{\mathsf{njp}} &= \left[v_{2}^{\mathsf{tra}}, v_{3}^{\mathsf{tra}}, v_{1}^{\mathsf{tra}}, v_{4}^{\mathsf{tra}}\right] \\
\mathbf{v}_6^{\mathsf{njp}} &= \left[v_{5}^{\mathsf{tra}}, v_{9}^{\mathsf{tra}}, v_{6}^{\mathsf{tra}}, v_{10}^{\mathsf{tra}}\right].
\end{aligned} \tag{11}
$$

D Details of Data augmentation

D.1 Intra-instance Augmentation

Skeletal augmentation. In our work, we employed skeletal augmentation, which includes random shear (multiply matrix Sh for three coordinate axes), random rotation (multiply matrix R for two random coordinate axes), random scaling (multiply matrix Sc for three coordinate axes), random skeletal flipping (swap the indices of the joints for joints with left and right pairs), random coordinate dropout (randomly remove one coordinate axis), random joint dropout (randomly remove a subset of the entire set of joints), and actor order permutation (randomly rearrange the order of actors) [14]. We did not utilize random temporal flip, random Gaussian noise, and random Gaussian blur [26] as they were found to degrade the performance of SkateFormer. Detailed information regarding the skeletal augmentation techniques employed in our study can be found in Table 11, where p_{aug} is application probability,

$$
\mathsf{Sh} = \begin{bmatrix} 1 & \mathsf{sh}_1 & \mathsf{sh}_2 \\ \mathsf{sh}_1 & 1 & \mathsf{sh}_2 \\ \mathsf{sh}_1 & \mathsf{sh}_2 & 1 \end{bmatrix}, \tag{12}
$$

$$
\mathsf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{13}
$$

and

$$
\mathsf{Sc} = \begin{bmatrix} \mathsf{sc}_1 & 0 & 0 \\ 0 & \mathsf{sc}_2 & 0 \\ 0 & 0 & \mathsf{sc}_3 \end{bmatrix}. \tag{14}
$$

Temporal augmentation. Fig. 7 illustrates our trimmed-uniform random sampling of frames with p portion. For this, a $p \times 100$ ($0.5 \leq p \leq 1$) percentage of the total input sequence length (T_{total}) is randomly determined before sampling. Given a p value, a start-frame index t_s is also randomly selected in the range of $0 \leq t_s \leq \lfloor T_{\mathsf{total}} \times (1-p) \rfloor$, and the end-frame index t_e is determined as $t_e = t_s + \lfloor T_{\mathsf{total}} \times p \rfloor$, thus constituting the selected interval $[t_s, t_e]$ which is further uniformly divided into total T sub-intervals. Then, as done in [12], total

Table 11: Details of skeletal augmentation.

Skeletal Augmentation	Hyperparameters for Each Dataset	
	NW-UCLA	NTU RGB+D, NTU RGB+D 120
Random Shear	$p_{\text{aug}} = 0$	$p_{\text{aug}} = 0.5$, $\text{sh}_i \in [-0.5, 0.5]$
Random Rotation	$p_{\text{aug}} = 1$, $\theta \in [-\pi/3, \pi/3]$	$p_{\text{aug}} = 0.5$, $\theta \in [-\pi/6, \pi/6]$
Random Scaling	$p_{\text{aug}} = 1$, $\text{sc}_i \in [0.5, 1.5]$	$p_{\text{aug}} = 0.5$, $\text{sc}_i \in [0.8, 1.2]$
Random Skeletal Flipping	$p_{\text{aug}} = 0$	$p_{\text{aug}} = 0.5$
Random Coordinate Dropout	$p_{\text{aug}} = 0$	$p_{\text{aug}} = 0.5$
Random Joint Dropout	$p_{\text{aug}} = 0$	$p_{\text{aug}} = 0.5$
Actor Order Permutation	$p_{\text{aug}} = 0$	$p_{\text{aug}} = 0.5$

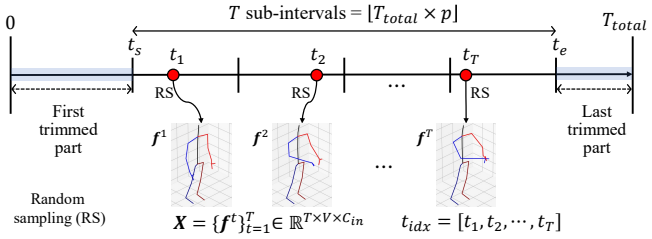

Fig. 7: The proposed trimmed-uniform random sampling of frames with p portion.

T frames are randomly selected with one random sampling in each sub-interval. When the selected interval $[t_s, t_e]$ is shorter than T ($T = 64$ in our case), we use linear interpolation to generate T frames by a built-in PyTorch [35] function `torch.nn.functional.interpolate`. In the training phase, we employed a random portion p (where $0.5 \leq p \leq 1$) and a random start-frame index t_s (where $0 \leq t_s \leq \lfloor T_{\text{total}} \times (1-p) \rfloor$). However, during inference, for consistency in results, we set $p = 0.95$ and $t_s = \lfloor T_{\text{total}} \times (1-p)/2 \rfloor$.

Skate-Embedding. Conventional sampling methods used in generating input skeleton sequences maintain either an absolute temporal gap (i.e., the fixed stride [7, 22, 75]) or a relative temporal gap (i.e., uniform random [10, 12]) between adjacent sampled frames. Therefore, embedding temporal indices, either absolutely or relatively, using learnable features has not posed a significant challenge. However, our sampling method relies on the random variable p to determine sub-intervals for sampling. Consequently, even for sequences generated from the same instance, the temporal gap between adjacent sampled frames and the sampling index of the first frame can differ significantly. This presents limitations in conveying temporal index information to the network through learnable features.

When $\lfloor T_{\text{total}} \times p \rfloor \geq T$, the sampled temporal indices are designated as $t_{\text{idx}} = [t_1, t_2, ..., t_T]$, as illustrated in Fig 7. As mentioned, in cases of $\lfloor T_{\text{total}} \times p \rfloor < T$, we perform linear interpolation across the entire index range $[t_1, t_2, ..., t_{\lfloor T_{\text{total}} \times p \rfloor}]$ to create $t_{\text{idx}} = [t'_1, t'_2, ..., t'_T]$. The following is the detailed equation for our fixed temporal index features TE based on traditional sinusoidal positional embeddings [55]:

Algorithm 1 Bone Length AdaIN

Input: Input skeleton sequence $\mathbf{X} \in \mathbb{R}^{T \times V \times C_{\text{in}}}$, (normalized) sampled temporal indices $\mathbf{t}_{\text{idx}} \in [-1, 1]^{T \times 1 \times 1}$, input true label $\mathbf{y}$, entire set of skeleton sequences $\{\mathbf{X}^n\}_{n=1}^N$, and predefined adjacency matrix $\mathbf{A}$.

Prepare reference sequence:

1: Choose the reference skeleton sequence (before sampling) $\mathbf{X}_{\text{f}}^{\text{b.s.}} \in \mathbb{R}^{T_{\text{total,f}} \times V \times C_{\text{in}}}$, which has same true label with $\mathbf{y}$ from $\{\mathbf{X}^n\}_{n=1}^N$.
2: Calculate the sampled temporal indices of $\mathbf{X}_{\text{f}}^{\text{b.s.}}$: $\mathbf{t}_{\text{idx,f}} = \text{int}((\mathbf{t}_{\text{idx}} + 1)/2 \times T_{\text{total,f}}) \in \mathbb{N}^{T \times 1 \times 1}$.
3: Sample the $\mathbf{X}_{\text{f}}^{\text{b.s.}}$ based on $\mathbf{t}_{\text{idx,f}}$ (denoted as $\mathbf{X}_{\text{f}} \in \mathbb{R}^{T \times V \times C_{\text{in}}}$).

Bone length calculation:

4: Calculate the bone of each skeleton sequence $\mathbf{B}, \mathbf{B}_{\text{f}} \in \mathbb{R}^{T \times V \times C_{\text{in}}}$ from $\mathbf{X}, \mathbf{X}_{\text{f}}$ based on $\mathbf{A}$.
5: Calculate the bone length of each skeleton sequence $\mathbf{L}, \mathbf{L}_{\text{f}} \in \mathbb{R}^{T \times V \times 1}$: $\mathbf{L} = ||\mathbf{B}||_2$ and $\mathbf{L}_{\text{f}} = ||\mathbf{B}_{\text{f}}||_2$.
6: Calculate the ratio of bone length: $\mathbf{r} = (\mathbf{L}_{\text{f}} + \text{eps})/(\mathbf{L} + \text{eps}) \in \mathbb{R}^{T \times V \times 1}$, where $\text{eps} = 10^{-6}$.

Bone length adaptation:

7: Adapt the bone length of input sequence: $\mathbf{B}_{\text{AdaIN}} = \mathbf{B} \times \mathbf{r}$.
8: Recover the skeleton sequence (joint) $\mathbf{X}_{\text{AdaIN}}$ from $\mathbf{B}_{\text{AdaIN}}$ based on $\mathbf{A}$.

Output: $\mathbf{X}_{\text{AdaIN}}$

$$\begin{aligned} \text{TE}[j, 2k] &= \sin(t_{\text{idx}}[j]/10000^{2k/C}) \\ \text{TE}[j, 2k+1] &= \cos(t_{\text{idx}}[j]/10000^{2k/C}), \end{aligned} \tag{15}$$

where $\text{TE} \in \mathbb{R}^{T \times C}$.

D.2 Inter-instance Augmentation

We propose a novel inter-instance augmentation technique called *Bone Length AdaIN*, drawing inspiration from Skeleton AdaIN [26]. Bone Length AdaIN enhances diversity among subjects with varying body sizes by exchanging the bone lengths across different subjects in different frame sequences, rather than within each frame sequence. Detailed information on the methodology can be found in Algorithm 1. We applied inter-instance augmentation with $p_{\text{aug}} = 0.2$.

E Self-Attention Analysis

E.1 Computational Complexity of Skate-MSA

When given the feature maps $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ of feature map $\mathbf{x}_{\mathsf{msa}}$, the computational complexity of the naive self-attention layer is $2(VT)^2(C/2)$:

$$\begin{aligned} \mathsf{Attn} = \mathsf{SoftMax}(\mathbf{Q}\mathbf{K}^{\mathsf{T}}) &\rightarrow [(VT)^2](\frac{C}{2}) \\ \mathsf{SA}(\mathbf{x}_{\mathsf{msa}}) = \mathsf{Attn} \cdot \mathbf{V} &\rightarrow [(VT)\frac{C}{2}](VT), \end{aligned} \tag{16}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{(T \times V) \times C/2}$.

Let's calculate the computation complexity for our proposed Skate-MSA. $\mathbf{x}_{\mathsf{msa}}$ is first split channel-wise ($\mathbf{x}^i_{\mathsf{msa}}$) and then partitioned into $\mathbf{x}^{i,\mathcal{P}}_{\mathsf{msa}}$. When considering the corresponding $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ for $\mathbf{x}^{i,\mathcal{P}}_{\mathsf{msa}}$, the computation complexity becomes $B \cdot 2(V'T')^2(C/8)$, where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{B \times (T' \times V') \times C/8}$. This can be easily computed in a manner similar to Eq. 16. The computation complexity for each Skate-Type is as follows:

$$\begin{aligned} \text{Skate-Type-1:} &(MK) \cdot 2(LN)^2\frac{C}{8} = 2(VT)^2\frac{C}{2}\left[\frac{1}{4} \cdot \frac{1}{MK}\right] \\ \text{Skate-Type-2:} &(ML) \cdot 2(KN)^2\frac{C}{8} = 2(VT)^2\frac{C}{2}\left[\frac{1}{4} \cdot \frac{1}{ML}\right] \\ \text{Skate-Type-3:} &(NK) \cdot 2(LM)^2\frac{C}{8} = 2(VT)^2\frac{C}{2}\left[\frac{1}{4} \cdot \frac{1}{NK}\right] \\ \text{Skate-Type-4:} &(NL) \cdot 2(KM)^2\frac{C}{8} = 2(VT)^2\frac{C}{2}\left[\frac{1}{4} \cdot \frac{1}{NL}\right], \end{aligned} \tag{17}$$

where $V = KL$ and $T = MN$. Therefore, by summing up all the complexities in Eq. 17, the overall complexity of Skate-MSA is as follows:

$$2(VT)^2\frac{C}{2}\left[\frac{1}{4}\left(\frac{1}{MK} + \frac{1}{ML} + \frac{1}{NK} + \frac{1}{NL}\right)\right]. \tag{18}$$

E.2 Skeletal-Temporal Positional Bias

To account for the characteristics of the temporal axis $\mathbf{t}$, we applied 1D relative positional bias: $\mathbf{B}^{\mathbf{t}}_h \in \mathbb{R}^{T' \times T'}$ [30] for the temporal splits, $\mathbf{t}^{\mathsf{local}}_m$ and $\mathbf{t}^{\mathsf{global}}_n$ in Eq. 6. Regarding the skeletal axis $\mathbf{v}$, for the skeletal split $\mathbf{v}^{\mathsf{njp}}_k$ in Eq. 3, due to the inconsistency between joints at the same position across partitions, no additional positional bias was applied: $\mathbf{B}^{\mathbf{v}}_h = \mathbb{1}^{V' \times V'}$. For the skeletal split $\mathbf{v}^{\mathsf{djp}}_l$ in Eq. 5, where elements at the same position always represent a consistent semantic part of the body, we utilized absolute positional bias: $\mathbf{B}^{\mathbf{v}}_h \in \mathbb{R}^{V' \times V'}$. The skeletal-temporal positional bias $\mathbf{B}_h$ is: $\mathbf{B}_h = \mathbf{B}^{\mathbf{t}}_h \otimes \mathbf{B}^{\mathbf{v}}_h \in \mathbb{R}^{T'V' \times T'V'}$, where $\otimes$ is a Kronecker product.

E.3 Additional Experiments on the Computational Complexity

Our SkateFormer comprises 8 SkateFormer blocks and 3 temporal downsampling layers. The 48× reduction in FLOPs was *theoretically calculated* from the first two SkateFormer blocks before downsampling, where the reduction effect is most pronounced. To clarify this, we additionally performed experiments in Table 12 in terms of measured FLOPs and memory usages in perspective of the attention layers *only*. As shown, the Skate-MSA is 38.87× less complex in terms of FLOPs for the first two attention layers.

Table 12: FLOPs and memory usage for attention layers.

Attention Types in attention layer	Total 8 attention layers		First two attention layers	
	FLOPs (M)	Memory (MB)	FLOPs (M)	Memory (MB)
$\mathbf{v}_k^{\mathsf{pjp}} + \mathbf{v}_l^{\mathsf{djp}} + \mathbf{t}_m^{\mathsf{local}} + \mathbf{t}_n^{\mathsf{global}}$	29.12	56.3	15.53	27.0
Naive self-attention	801.79 (×27.53)	1530.0 (×27.18)	603.65 (×**38.87**)	1152.0 (×**42.67**)

F Details of Implementation

F.1 Hyperparameters

Here is a more detailed description of the hyperparameters we used. We employed the AdamW optimizer [32] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay value of 0.1. For the cosine-annealing scheduler [31], we set warmup learning rate to 10^{-7}, minimum learning rate to 10^{-5}, base learning rate to 10^{-3}, warm-up epochs to 25, and utilized linear warm-up strategy. To prevent overfitting in the network, we set the attention map dropout ratio to 0.5 and the drop path ratio to 0.2. We did not use dropout for classification head and linear layers. The number of heads H was consistent across all SkateFormer Blocks, set at 32. The expansion ratio of FFN was set to 1 for the NW-UCLA [56] dataset and 4 for the NTU RGB+D [41] and NTU RGB+D 120 [27] datasets. We employed the GELU activation function in our model. Furthermore, we employed a total of $R = 8$ SkateFormer Blocks, applying temporal downsampling after every 2 blocks. The kernel size for the T-Conv layers was set to $k = 7$. For the performance evaluation, we compare our SkateFormer with other methods in terms of the top-1 action recognition accuracy for the test data sets.

F.2 Overall Architecture

The overall architecture of SkateFormer and the output shape of feature maps produced by each module are presented in Table 13.

Table 13: The details of SkateFormer.

	Mudule	Output Shape
Input		$64 \times V \times 3$
Projection	Linear 1	$64 \times V \times 6$
	Linear 2	$64 \times V \times 9$
	Linear 3	$64 \times V \times 96$
Stage 1	SkateFormer Block 1	$64 \times V \times 96$
	SkateFormer Block 2	$64 \times V \times 96$
Downsampling	T-Conv ($\downarrow_2$) 1	$32 \times V \times 96$
Stage 2	SkateFormer Block 3	$32 \times V \times 192$
	SkateFormer Block 4	$32 \times V \times 192$
Downsampling	T-Conv ($\downarrow_2$) 2	$16 \times V \times 192$
Stage 3	SkateFormer Block 5	$16 \times V \times 192$
	SkateFormer Block 6	$16 \times V \times 192$
Downsampling	T-Conv ($\downarrow_2$) 3	$8 \times V \times 192$
Stage 4	SkateFormer Block 7	$8 \times V \times 192$
	SkateFormer Block 8	$8 \times V \times 192$
Head	Pool	$1 \times 1 \times 192$
	Linear (classification)	$1 \times 1 \times N_c$
Output		$1 \times 1 \times N_c$

F.3 Modules

Details of G-Conv. Let us denote $\mathbf{G}$ as learnable parameters, where $\mathbf{G} \in \mathbb{R}^{V \times V \times H/4}$. The input to G-Conv, denoted as $\mathbf{x}_{\mathsf{gc}}$, is split into $H/4 (= 32/4 = 8)$ features in a channel-wise manner:

$$[\mathbf{x}_{\mathsf{gc}}^1, ..., \mathbf{x}_{\mathsf{gc}}^{H/4}] = \mathsf{Split}(\mathbf{x}_{\mathsf{gc}}), \tag{19}$$

where $\mathbf{x}_{\mathsf{gc}} \in \mathbb{R}^{T \times V \times C/4}$. Each $\mathbf{x}_{\mathsf{gc}}^h$ undergoes matrix multiplication with its corresponding $\mathbf{G}_h = \mathbf{G}[:, :, h]$, and the results are channel-wise concatenated to form the output of G-Conv:

$$\mathsf{G\text{-}Conv}(\mathbf{x}_{\mathsf{gc}}) = \mathsf{Concat}(\mathbf{G}_1 \cdot \mathbf{x}_{\mathsf{gc}}^1, ..., \mathbf{G}_{H/4} \cdot \mathbf{x}_{\mathsf{gc}}^{H/4}), \tag{20}$$

where $\mathbf{G}_h \in \mathbb{R}^{V \times V}$. Through this process, we obtain an inductive bias with respect to skeletal position.

Details of FFN. The FFN of the SkateFormer Block is expressed as follows: $\mathbf{x} \leftarrow \mathbf{x} + \mathsf{Linear}(\mathsf{Act}(\mathsf{Linear}(\mathsf{LN}(\mathbf{x}))))$, where $\mathbf{x} \in \mathbb{R}^{T \times V \times C}$. If we denote the expansion ratio as e, the first linear layer in the FFN expands features from C channels to a higher-dimensional $e \times C$ channels, and the second linear layer squeezes them back to C channels from $e \times C$ channels. For the NW-UCLA dataset, we set $e = 1$, while for the NTU RGB+D and NTU RGB+D 120 datasets, we used $e = 4$.

Table 14: List of symbols.

	Symbol	Description
Constant	T_{total}	The total input sequence length (Varies from one instance to another)
	T	The number of sampled frames ($T = 64$ in our case)
	V	The number of total joints
	C_{in}	The dimensionality of data representing a single joint ($C_{\text{in}} = 3$ in our case)
	N_c	The number of classes
Hyper-parameter	C	The dimensionality of feature map
	H	The number of total heads
	R	The number of total SkateFormer Blocks, $R_1 + R_2 + R_3 + R_4$
	K	The number of neighboring joint partitions
	L	The number of distant joint partitions
	M	The number of local frame partitions
	N	The number of global frame partitions
Input	$\mathbf{X}$	Input sampled skeleton sequence ($\mathbf{X} \in \mathbb{R}^{T \times V \times C_{\text{in}}}$)
	$\mathbf{f}^t$	t-th frame of input skeleton sequence ($\mathbf{f}^t \in \mathbb{R}^{V \times C_{\text{in}}}$)
	t_{idx}	The sampled temporal indices ($t_{\text{idx}} \in \mathbb{R}^T$)
Output	$\hat{\mathbf{y}}$	The outcome of SkateFormer ($\hat{\mathbf{y}} \in \mathbb{R}^{N_c}$)
	$\mathbf{y}$	The true label ($\mathbf{y} \in \mathbb{R}^{N_c}$)
Skate-MSA	$\boldsymbol{\mathcal{P}}_i$	The i-th Skate-Type partition operation
	$\boldsymbol{\mathcal{R}}_i$	The i-th Skate-Type reverse operation
	$\mathbf{v}$	The skeletal axis of feature map $\mathbf{x}$
	$\mathbf{t}$	The time axis of feature map $\mathbf{x}$
	$\mathbf{v}_k^{\text{njp}}$	The k-th neighboring joint partition, $\mathbf{v} = \{\mathbf{v}_k^{\text{njp}}\}_{k=1}^K$
	$\mathbf{v}_l^{\text{djp}}$	The l-th distant joint partition, $\mathbf{v} = \{\mathbf{v}_l^{\text{djp}}\}_{l=1}^L$
	$\mathbf{t}_m^{\text{local}}$	The m-th local frame partition, $\mathbf{t} = \{\mathbf{t}_m^{\text{local}}\}_{m=1}^M$
	$\mathbf{t}_n^{\text{global}}$	The n-th global frame partition, $\mathbf{t} = \{\mathbf{t}_n^{\text{global}}\}_{n=1}^N$
Feature	$\mathbf{x}$	The input for SkateFormer Block ($\mathbf{x} \in \mathbb{R}^{T \times V \times C}$)
	$\mathbf{x}_{\text{gc}}$	The input for G-Conv ($\mathbf{x}_{\text{gc}} \in \mathbb{R}^{T \times V \times \frac{C}{4}}$)
	$\mathbf{x}_{\text{tc}}$	The input for T-Conv ($\mathbf{x}_{\text{tc}} \in \mathbb{R}^{T \times V \times \frac{C}{4}}$)
	$\mathbf{x}_{\text{msa}}$	The input for Skate-MSA ($\mathbf{x}_{\text{msa}} \in \mathbb{R}^{T \times V \times \frac{C}{2}}$)
	$\mathbf{x}_{\text{msa}}^i$	The input for MSA of i-th Skate-Type ($\mathbf{x}_{\text{msa}}^i \in \mathbb{R}^{T \times V \times \frac{C}{8}}$)
	$\mathbf{x}_{\text{msa}}^{i,\boldsymbol{\mathcal{P}}}$	The partitioned $\mathbf{x}_{\text{msa}}^i$, $\boldsymbol{\mathcal{P}}_i(\mathbf{x}_{\text{msa}}^i)$