Autoregressive Adaptive Hypergraph Transformer for Skeleton-based Activity Recognition

Abhisek Ray[1] Ayush Raj[1] Maheshkumar H. Kolekar[1]
[1]Indian Institute of Technology Patna, India
rayabhisek0610@gmail.com, {ayush_2001ee10, mahesh}@iitp.ac.in

Abstract

*Extracting multiscale contextual information and higher-order correlations among skeleton sequences using Graph Convolutional Networks (GCNs) alone is inadequate for effective action classification. Hypergraph convolution addresses the above issues but cannot harness the long-range dependencies. The transformer proves to be effective in capturing these dependencies and making complex contextual features accessible. We propose an **Autoreg**ressive **Ad**aptive **H**yper**G**raph Trans**former** (**AutoregAd-HGformer**) model for in-phase (autoregressive and discrete) and out-phase (adaptive) hypergraph generation. The vector quantized in-phase hypergraph equipped with powerful autoregressive learned priors produces a more robust and informative representation suitable for hyperedge formation. The out-phase hypergraph generator provides a model-agnostic hyperedge learning technique to align the attributes with input skeleton embedding. The hybrid (supervised and unsupervised) learning in AutoregAd-HGformer explores the action-dependent feature along spatial, temporal, and channel dimensions. The extensive experimental results and ablation study indicate the superiority of our model over state-of-the-art hypergraph architectures on the NTU RGB+D, NTU RGB+D 120, and NW-UCLA datasets. Find the code Here.*

1. Introduction

In recognizing various human actions, less attention has been paid to the body skeleton compared to other modalities [27, 33, 45] like appearance, optical flow, and depth [41]. Nowadays, skeleton-based action recognition has gained popularity due to its immutable nature towards viewpoint variation, illumination changes, and background clutter. Reduced computational cost and background adaptation widen its application in surveillance, smart security, healthcare, and Human-Computer Interaction (HCI) [31, 40].

Earlier, the non-euclidean graph geometry of skeleton sequences related to human actions is analyzed through various fully supervised architectures such as image-based Convolutional Neural Networks (CNNs) [13, 15, 26], sequence-based Recurrent Neural Networks (RNNs) [17, 22, 43, 47], graph-based Graph Convolutional Networks (GNNs) [5, 8, 14, 25, 41], and attention-based transformers [10, 18–20, 39]. Among them, GCN-based architecture is widely adopted because the inductive biases present in CNNs or RNNs are unable to capture the non-euclidean joint-bone attributes of the human skeleton.

The ST-GCN [41], a pioneering architecture for graph geometry, uses a fixed sampling method for selecting the neighborhood to perform convolution operations on skeleton sequences. However, the predefined adjacency matrix, which represents the natural connections of a human skeleton, fails to address the high-order correlations between the farthest skeleton nodes and is unable to capture the multi-scale semantic information. To get rid of the above drawbacks, adaptive graph topologies are introduced, combining predefined adjacency graph topology with either learnable node topology [23] or learnable channel topology [5] to fetch action-dependent skeleton features. The above approaches manage to aggregate these features to alleviate the performance, but the scale of effectiveness still needs to be higher to handle the diversity in state-of-the-art datasets. While several architectures [5, 8, 14, 37] have emerged to address these challenges, their effectiveness has been only partially realized.

Hypergraphs over graph topology: Hypergraphs [11, 35, 48] offer significant advantages over traditional graphs in skeleton-based activity recognition by capturing higher-order relationships among joints, which are crucial for actions like "waving," where multiple joints (shoulder, elbow, wrist) need to be considered simultaneously. They represent multi-scale contextual information, allowing for a detailed understanding of local and global patterns in actions such as "running" or "walking," where limb and full-body coordination are both essential. Hypergraphs are also more robust to noise by emphasizing critical joints, which helps in actions like "sitting down," where irrelevant joint movements can mislead simpler models. Enhanced feature fusion in

hypergraphs aids in recognizing actions like "clapping" by effectively combining spatial and temporal dynamics across multiple joints. They also adapt dynamically, adjusting joint importance throughout actions like "jumping," where the focus shifts between legs and arms during the sequence. In complex activities such as "picking up and throwing an object," hypergraphs excel by capturing the intricate interdependencies among multiple body parts involved in the motion. These capabilities make hypergraphs particularly powerful for accurately recognizing a wide range of skeleton-based actions.

Two primary strategies for generating hyperedges to form a hypergraph from skeletal sequences are: (i) fixed-generated hypergraph [46] and (ii) dynamically generated hypergraph [11, 36, 38] techniques. Since the set of nodes in each hyperedge is defined by one-to-one correlations between body joints within a specific action frame, a fixed hypergraph cannot effectively capture the dynamic, action-dependent contextual features. While dynamically generated hypergraphs can partially incorporate action-dependent attributes [36, 38], the frequent transitions of the hypergraph inside the descriptors disrupt the feature distribution, thereby limiting its effectiveness for classification. However, we propose in-phase and out-phase hypergraph techniques to boost representation for intra-batch and inter-batch embeddings. By applying vector quantization with autoregressive learned priors, the input hypergraph is discretized to produce the in-phase hypergraph inside the feature extractor (encoder). It prevents frequent node transition between hyperedges and boosts robustness and dynamic adaptation by providing better representation. Similarly, we also designed a decoder that generates model-agnostic out-phase hypergraphs outside the extractor. Here, the hypergraph is adaptively restructured in response to action-specific features. It takes hypergraph-attentive spatiotemporal features from the encoder to generate an action-dependent model-independent hypergraph for the next iteration.

We adopt a unique transformer design that analyzes individual features of joint and hyperedge along with their mutual semantics. The overall abstraction behind our model implementation is shown in Fig. 1. The contributions of the proposed AutoregAd-HGformer can be summarized as:

1. We propose an autoregressive adaptive HyperGraph Transformer (AutoregAd-HGformer) model that introduces a unique transformer-implemented hypergraph architecture to skeleton-based action sequences and mutates hyperedge configuration adaptively to avail more diverse features for recognition.

2. We propose two unique hypergraph generation techniques to produce in-phase and out-phase hypergraphs for discrete and continuous feature alignment, respectively. Leveraging both model-dependent and model-agnostic hypergraphs, AutoregAd-HGformer preserves intra-batch and inter-batch relationships, thereby enhancing the richness of hypergraph representations.

3. The hybrid learning (supervised and self-supervised) in AutoregAd-HGformer explores the action-dependent feature along spatial, temporal, and channel dimensions.

4. The extensive experimental results and ablation study indicate the superiority of our model over state-of-the-art hypergraph architectures on NTU RGB+D, NTU RGB+D 120, and NW-UCLA datasets.

2. Related Works

Numerous graph architectures, including graph convolution networks, auto-encoders, transformers, and hypergraph convolution networks, address non-Euclidean data geometry. Our hypergraph framework leverages multi-scale semantics and non-linear contextual features using hypergraph convolution and transformers. These methods play a crucial role in developing the novel hypergraph transformer-based AutoregAd-HGformer.

2.1. Graph Transformer Methods

To move beyond handcrafted traversal rules or graph topologies, DSTA-Net introduces a transformer-based architecture using decoupled spatiotemporal self-attention for skeleton-based action and gesture recognition [24]. Depending on the sampling rate, the temporal stream is split into fast and slow streams. SAN [9] presents three network variants using multi-head self-attention on encoded or fused skeleton features of action sequences. Plizzari et al. [21] developed a spatial-temporal transformer for skeleton-based activity recognition, learning intra-frame interaction and inter-frame correlation through spatial and temporal convolutions and transformers. STSA [1] decouples spatial and temporal self-attention in parallel with a combined feed-forward network. Zhang et al. [44] designed STST, a sequential block of special and directional temporal transformers, for modeling skeleton-based action sequences. FG-STformer [10] emphasizes local joint and global body part features via self- and cross-attention layers. For skeleton-based human interaction recognition, [20] proposes a semantic partition module generating Body-Part-Time sequences fed to IGFormer, which enhances graph representation with cross-attention. Hue et al. [18] apply a similar approach by dividing skeleton action sequences into upper-lower and hand-foot joints. All frameworks above use supervised learning with labeled skeletal action sequences. The scarcity of labeled data in human action

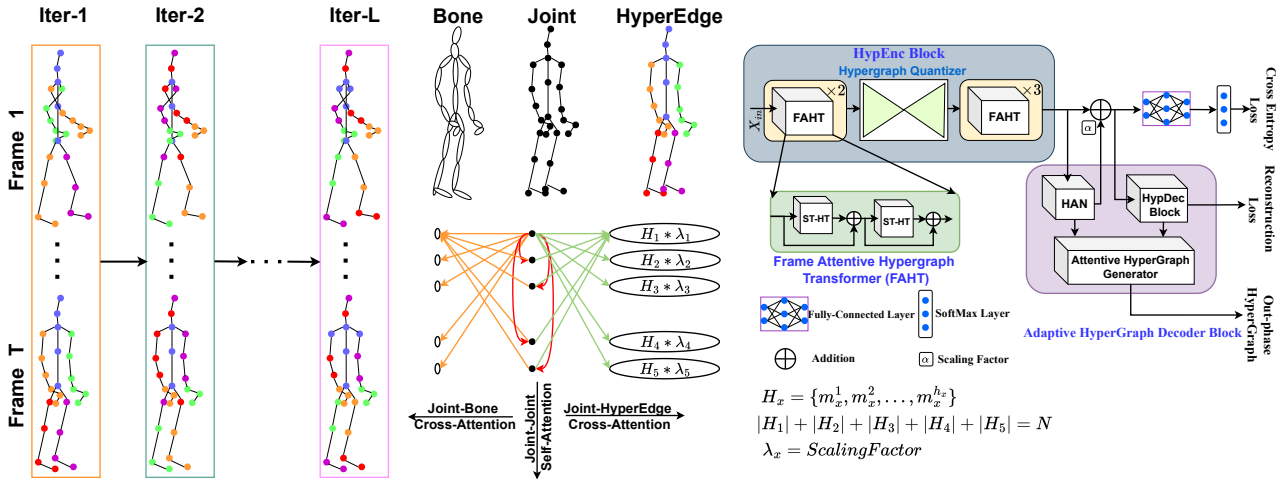

Figure 1. Model abstraction. Model-agnostic iterative hypergraph (left), various attention (middle) and AutoregAd-HGformer (right)

recognition limits the full potential of transformers. Recently, self-supervised transformer architectures [19, 39] have been introduced, using masking techniques to improve performance. Wu et al. [39] proposed SkeletonMAE, employing a masking and reconstruction pipeline with a self-supervised encoder-decoder transformer. MAMP [19] introduces motion-aware masking based on motion intensity for motion predictions of joint embeddings from skeletal action sequences, utilizing multi-head self-attention in the encoder and decoder.

2.2. Hypergraph Convolution Methods

Wei et al. introduced DHGCN, a hypergraph convolution network for skeleton-based action recognition using dynamic topology and dynamic joint weights [38]. Dynamic topology groups nodes based on relative features, and dynamic joint weights assign importance-based weights to nodes. However, the hypergraph's dynamic variation in each extraction layer can disrupt the deep features fed to the classification head. Concurrently, Hao et al. proposed Hyper-GNN, a hypergraph skeleton model that captures multiscale spatiotemporal intelligence and higher-order dependencies among skeleton sequences [11]. While it reduces noise from uncorrelated joints, it does not address the limitations of [38]. In both [38] and [11], hyperedge construction is based on the moving distance between consecutive frames. Selective-HCN [48] uses selective-scale hypergraph convolution (SHC) and selective-frame temporal convolution (STC) to aggregate spatiotemporal features. SHC captures and fuses multiscale information selectively, while STC extracts keyframes for temporal features. However, Selective-HCN cannot extract action correspondence features due to a fixed hypergraph and struggles with distinguishing similar actions at different paces. To address separate learning of spatiotemporal information, DST-HCN [36] dynamically and parallelly learns using dynamic hypergraph convolution and an information fusion block, but it still exhibits drawbacks similar to [38] and [11].

The AutoregAd-HGformer uses hypergraph convolution and the hypergraph transformer along with in-phase and out-phase hypergraph generation in the feature extraction unit to mitigate the above issues and effectively derive multiscale semantics and efficient motion features.

3. Proposed Method

As shown in Fig. 1, the proposed AutoregAd-HGformer comprises three functional blocks: (i) hypergraph encoder, (ii) adaptive hypergraph decoder, and (iii) classifier. A fully connected and a softmax layer are combined to form the classifier block. The other two blocks are discussed below:

3.1. Hypergraph Encoder

The Hypergraph Encoder (HypEnc) block comprises a stack of five Frame Attentive Hypergraph Transformer (FAHT) units divided between two groups in a 2:3 ratio. The output embedding of the first group is passed through a hypergraph quantizer to find an autoregressive discrete hypergraph for the second group. The output embedding E for each node of the HypEnc block can be expressed as:

$$E = \text{HypEnc}(\mathbf{A}, X, \mathbf{H^n}, \mathbf{H_w}^{\mathbf{n}}), \quad (1)$$

where $\mathbf{A} \in \{0,1\}^{V\times V}$ is the adjacency matrix, $\mathbf{X} \in \mathbb{R}^{N\times M\times V\times T\times C}$ represents the input feature embedding, $\mathbf{H}^n \in \{0,1\}^{V\times E_h}$ is the incidence matrix, $\mathbf{H_w}^n \in \mathbb{R}^{E_h\times E_h}$ denotes the hyperedge weight matrix, and $\mathbf{E} \in \mathbb{R}^{NM\times V\times T'\times C'}$ corresponds to the output embedding. The parameters n, N, M, V, T, C, E_h, T', and C' refer to the iteration number, batch size, number of individuals per frame, skeletal joint count, frame count per action, input channel count, number of hyperedges, number of output frames, and output channel count, respectively. Following in-phase and out-phase hyperedge calculations, the updated hypergraph is propagated to the next iteration every time. The FAHT unit in each group comprises the Spatio-Temporal Hypergraph Transformer (ST-HT) unit and the Spatiotemporal Attentive Hypergraph Transformer (STA-HT) unit.

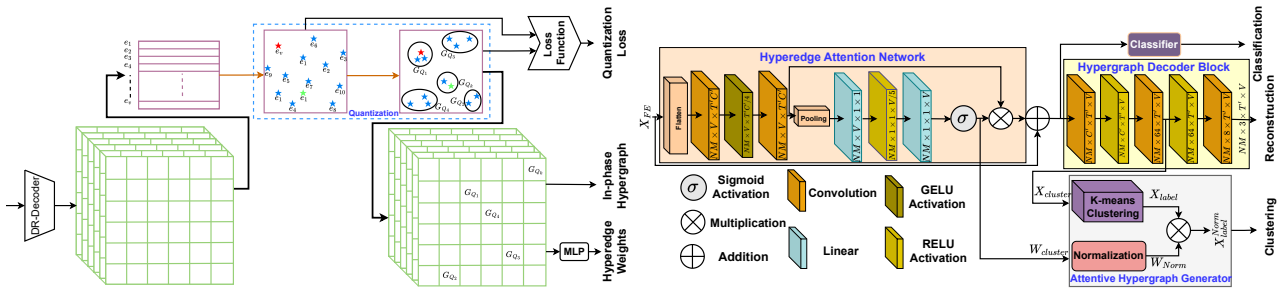

Figure 2. Proposed framework for autoregressive in-phase hypergraph quantizer (left) and adaptive hypergraph decoder (right).

ST-HT: The **H**yper**G**raph **C**onvolution (HG_C) is applied to find hypergraph embedding that subsequently passes to the transformer unit to calculate cross-attention. The hypergraph convolution is defined by

$$H_f = \lambda_1 \mathrm{HG}_C(X, \mathbf{H^n}, \mathbf{H^n_w}) + \lambda_2 \mathrm{HG}_C(X, \mathbf{H^n_q}, \mathbf{H^n_{qw}}). \quad (2)$$

$\mathbf{Hq^n} \in \{0,1\}^{N*M\times V\times E_h}$ and $\mathbf{H^n_{qw}} \in \mathbb{R}^{N*M\times E_h\times E_h}$ are the quantized in-phase hypergraph and its weight, respectively. H_n and H^n_w are the out-phase hypergraph and its weight from the previous iteration, respectively. For the first two FAFT units, $\mathbf{H^n_q} = \mathbf{H^n}$ and $\mathbf{H^n_{qw}} = \mathbf{H^n_w}$ as quantized hypergraph is calculated after the second unit as shown in Fig. 2. The equations

$$sa^l_{i,j} = q^l_i {k^l_j}^T, ca_{h}{}^l_{i,j} = q^l_i {H^l_f}_j{}^T, ca_r{}^l_{i,j} = q^l_i {P^l_{i,j}}^T \quad (3)$$

represent the Joint-to-Joint self-attention, Joint-to-Hyperedge cross-attention and Joint-to-Bone cross-attention for the l^{th} head, respectively. The q and k are linearly transformed query and key vectors. $P^l \in \mathbb{R}^{N\times N\times(C'/h)}$ is the bone attributes of the skeleton sequences. All these attention embeddings are aggregated and multiplied with value (v) vector by the following equation:

$$Y^l_i = \sum_{j=1}^{N} \frac{a^l_{t_{i,j}}}{\sum_j a^l_{t_{i,j}}} v^l_j. \quad (4)$$

Here $a^l_{t_{i,j}}$ is the aggregated representation of l^{th} head. At the end, all the heads $\{Y^0_i, Y^1_i, ..., Y^{h-1}_i\}$ are concatenated to form the final output Y_i for the ith node.

STA-HT: All the functional units in STA-HT are the same as the ST-HT block, except temporal attention is applied to hypergraph features in channel dimensions to recognize the importance of each frame. Details can be found in the supplementary material.

Hypergraph Quantizer: As shown in Fig. 2, we follow the autoregressive vector quantization method [28] for discrete in-phase hypergraph generation. To escape from the curse of dimensionality [3], the output embeddings $f \in \mathbb{R}^{V\times D}$ from the second FAHT unit is passed through a decoder block to get an intermediate low-dimensional vector $E \in \mathbb{R}^{V\times d}$. The decoder used here is very similar to the decoder used in the adaptive hypergraph decoder block (sec. 3.2). The embedding E is passed through a discretization bottleneck and subsequently maps to the nearest hyperedge Q_j with the help of autoregressive trainable vectors, which served as codebook $\{G_{Q_1}, ..., G_{Q_K}\}$. The assignment operator for these discrete input-independent hyperedges is defined by

$$he_i = argmin_j \|E_i - Q_j\|_2, \quad (5)$$

where $|he_i| = K$ denotes the hyperedge. The quantized vector Q_j, also referred to as an in-phase hypergraph H_q, is passed through an MLP layer to get the weights of the hypergraph H_{q_w}. The objective of learning these vectors is similar to clustering the nodes during hypergraph formation. As quantization is a lossy and non-differentiable process, the gradient is calculated over the quantizer input instead of intermediate discrete embedding [2].

3.2. Adaptive Hypergraph Decoder

The features E_{enc} from the hypergraph encoder block are brought for classification and out-phase hypergraph generation after passing through a Hyperedge Attention Network (HAN) as shown in Fig. 2. The output embeddings $A_t \in \mathbb{R}^{NM\times V\times T'\times C'}$ from HAN are passed through the decoder to reconstruct the skeleton sequences and provide low-dimensional features $E_c \in \mathbb{R}^{NM\times V\times d}$ to the adaptive hypergraph generator, where the out-phase hypergraph is prompted. We take $d \ll C'$ due to the "curse of dimensionality" during clustering. We take d (hyperparameter) as 8 for a C' of 128. The output from the second layer of the decoder is considered to calculate the model-agnostic out-phase hypergraph. The following equations describe the above process.

$$A_t = \mathrm{HAN}(E_{enc}) \quad (6)$$

$$E_c = \mathrm{HypDec}_s(\mathrm{GAP}_{\mathrm{time}}(E_{enc} + \alpha A_t), \mathbf{A}) \quad (7)$$

$$Emb_c = \mathrm{Norm}(\mathrm{GAP}_{\mathrm{batch}}(E_c)) \quad (8)$$

Here, $\mathrm{GAP}_{\mathrm{time}}$, and $\mathrm{GAP}_{\mathrm{batch}}$ represent the normalization, global average pooling in the temporal dimension, and global average pooling across the total number of skeletons, respectively.

Hyperedge Attention Network (HAN): For hyperedge attention, we compute the weights from the output embeddings of the HypEnc block using a squeeze-excitation network [4].

$$E_2 = \text{Conv}_1(\text{GELU}(\text{Conv}_2(E_{enc}))) \quad (9)$$

We use dot hypergraph attention as shown in the equation:

$$A_t = attn \odot E_2, \text{where} \quad (10)$$

$$attn = \sigma(F^{node}_{Lin_2}(\text{ReLU}(F^{node}_{Lin_1}(E_3)))), \quad (11)$$

$$E_3 = \text{GAP}_{T'C'}(E_2). \quad (12)$$

Here, $\text{GAP}_{T'C'}$, $F^{node}_{Lin_1}$, $F^{node}_{Lin_2}$, and σ represent the global average pooling along time and channel, two linear layers, and sigmoid activation layers, respectively.

Hypergraph Decoder (HypDec): The combined residual and attentive features from HAN are passed through the 5-layered decoders, where each layer performs the following hypergraph convolution.

$$\mathbf{Y} = \mathbf{D}^{-1/2}\mathbf{A_1}\mathbf{D}^{-1/2}\mathbf{X_1}\mathbf{W}. \quad (13)$$

Here, $\mathbf{X_1} \in \mathbb{R}^{V_1 \times C_1}$, $\mathbf{A_1} \in \{0,1\}^{V \times V}$, D, and $\mathbf{W} \in \mathbb{R}^{C_1 \times C_2}$ represent the decoder input, adjacency matrix, degree matrix, and weight matrix of the hypergraph, respectively. The output from the third layer is taken for out-phase hypergraph generation. The output feature from the decoder is used for graph reconstruction, which advocates self-supervised learning.

Attentive Hypergraph Generator: K-means clustering is applied to the intermediate output of the decoder to find hyperedges. The weights from HAN are normalized to calculate the importance of each hyperedge. The model-agnostic hypergraph can be expressed by

$$\mathbf{H^{l+1}_{i,j}} = \begin{cases} 1 & \text{if joint } i \text{ is assigned to cluster } j \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

and their weights are defined by

$$\mathbf{H^{l+1}_w} = diag(W_1, W_2, ..., W_T), \text{where} \quad (15)$$

$$W_t = (\mathbf{H^{l+1}_t} A_{t_{joints}}) / \sum_{i=1}^{N} A_{t_{joints\,i}} \quad (16)$$

Here, $\mathbf{H^{l+1}_{i,j}}$ and $\mathbf{H^{l+1}_w}$ are the hyperedges and their respective weights for the next iteration.

The process can be better understood from the algorithm 1 given below. The reconstruction loss is introduced as regularization, which avoids forgetting lower-level dependencies in the embeddings E.

Algorithm 1 Forward propagation of training process

for $n \leftarrow 1$ to $total\_iteration$ **do**
  if $n == 1$ **then**
    $\mathbf{H^n} \leftarrow$ RandomAllocationOfNodesToHyperedges
    $\mathbf{H^n_w} \leftarrow I$(Identity Matrix)
  end if
  $E_{enc} \leftarrow$ HypEnc($\mathbf{A}, X, \mathbf{H^n}, \mathbf{H^n_w}$)
  $A_t, attn \leftarrow$ HAN(E_{enc}), $E_f = E_{enc} + \alpha A_t$
  $rec \leftarrow$ HypDec($\text{GAP}_{\text{time}}(E_f), \mathbf{A}$),$E_c \leftarrow$ HypDec($\text{GAP}_{\text{time}}(E_f), \mathbf{A}$)
  Feed E_f to classification and reconstruction head
  Update hypergraph for next iteration
  $\mathbf{H^{n+1}}$, $\mathbf{H^{n+1}_w} \leftarrow$ ATTENTIVE HYPERGRAPH GENERATOR($attn, E_c$)
end for
procedure ATTENTIVE HYPERGRAPH GENERATOR($attn, E_c$)
  Refer to section 4.2
  return $\mathbf{H^{n+1}}$, $\mathbf{H^{n+1}_w}$
end procedure

4. Loss Functions

Our model has encountered three losses due to vector quantization, classification, and reconstruction.

Cross-entropy loss: It is the measure of dissimilarity between the true distribution y of labels and the predicted distribution $\hat{y}$. For a batch of N skeleton sequences of C various actions, the average cross-entropy loss is given by

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_c^{(n)} \log(\hat{y}_c^{(n)}) \quad (17)$$

Reconstruction loss: It is the measure of the discrepancy between the input X and reconstructed output distributions $\hat{X}$, encouraging the model to accurately replicate the input skeleton sequence. The reconstruction loss is defined by

$$\mathcal{L}_{\text{rec}} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{c=1}^{C}\left\| X^{(n)}_{t,j,c} - \hat{X}^{(n)}_{t,j,c} \right\|^2 \quad (18)$$

where N, T, J, and C denote the batch size, number of time frames, number of skeletal joints, and number of channels, respectively. $\|\cdot\|$ denotes the L_2 norm. The decoder helps regularize the generation of hypergraphs. It tries to restrict the embeddings from overfitting to high-level features and sustain some part of low-level features. As we use the similar decoder during both in-phase and out-phase hypergraph generations, two reconstruction losses $\mathcal{L}_{\text{rec1}}$ and $\mathcal{L}_{\text{rec2}}$ are introduced to the total cost objective.

Quantization loss: Due to the lossy quantization operation, a new loss function is added to the total cost objective, which can be defined by

$$\mathcal{L}_{\text{quant}} = (1/V)\sum_{i=1}^{V} \|Q_{he_i} - sg(E_i)\|_2^2. \quad (19)$$

"sg" denotes the non-updated stop gradient operator calculated only during the forward pass, as it has zero gradients in backpropagation. We follow the straight-through estimator approach [28] for backpropagation.

The total loss encountered during the model training is represented by

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \beta_1\mathcal{L}_{\text{rec1}} + \beta_2\mathcal{L}_{\text{rec2}} + \beta_3\mathcal{L}_{\text{quant}}. \quad (20)$$

Here, β_1, β_2, and β_3 are the scaling factors (hyperparameters) to be optimized during model training.

5. Experiments

5.1. Datasets

NTU RGB+D 60 [22]: It contains 56,880 sequences of 25 joints annotated for 60 action classes. The action sequences of 40 subjects are captured from three different Kinect camera angles. Cross-view (X-View) and cross-subject (X-Sub) are the protocols followed during the evaluation.

NTU RGB+D 120 [16]: By adding 57,367 skeletons to NTU RGB+D 60, the extended dataset NTU RGB+D 120 is created by adding 60 more classes. The scenes feature 106 performers in 32 different configurations. It also uses two protocols: cross-subject (X-Sub) and cross-setup (X-Set).

NW-UCLA [30]: There are 1494 sequences in the NW-UCLA dataset that show ten distinct actions. Every sequence has twenty joint annotations for added detail. Ten people perform these actions, captured by three Kinect cameras from different perspectives.

5.2. Implementation details

The model is trained over 140 epochs using standard cross-entropy loss. The learning rate is initialized at 0.025 and decreased by a factor of 0.1 at the 110th and 120th epochs. We use an SGD optimizer with a Nesterov momentum value of 0.9 and a weight decay of 0.0004 during training. For NTU RGB+D and NTU RGB+D 120, a batch size of 64 is employed, with each sample resized to 64 frames, and data preprocessing follows the method described in [42]. Northwestern-UCLA experiments use a batch size of 16 and adhere to the data preprocessing steps outlined in [5]. The frame rate of each action sequence and the number of hyperedges are set to 64 and 5, respectively. We set the number of hyperedges, E_h or K, as 5. The model architecture consists of 10 layers with 216 hidden channel dimensions for all experiments. The experiments are conducted using Python 3.10.11 based on the PyTorch 2.0.1 deep learning framework on an Ubuntu 20.04.2 machine equipped with an A100-PCIE-40GB GPU enabled with Nvidia CUDA 10.1.243 and CuDNN 8.1.0.

5.3. Ablation Analysis

In this section, we analyze the impact of all attention techniques, hypergraph generation modules, scaling factors, channel count, and layer count of the decoder. To conduct the ablation study, we exclusively use the X-sub benchmark of the NTU RGB+D dataset.

Table 1. Impact of various attention in transformer block.

Model	Joint - Joint	Joint - Hypergraph	Joint - Bone	Accuracy in (%)
Base-line+	✓			93.09
			✓	93.18
		✓	✓	93.68
	✓	✓		93.77
	✓		✓	93.80
	✓	✓	✓	94.15

Table 2. Impact of different units in AutoregAd-HGformer.

Model	out-phase Hypergraph	in-phase Hypergraph	Hypergraph Attention	Accuracy in (%)
Base Model +		✓	✓	93.84
	✓			93.65
	✓	✓		93.88
	✓		✓	93.79
	✓	✓	✓	94.15

Impact of different attention used in transformer block: As discussed in section 3.1, we have used one self-attention and two cross-attention in the proposed AutoregAd-HGformer. Each carries different significance and weight in the output performance. As shown in Table 1, we calculate the model performance (accuracy in %) of various attention in conjunction with the baseline model. The proposed model performs best when all attention is placed on a single framework.

Impact of different units in AutoregAd-HGformer: Apart from the feature attention discussed in the previous paragraph, we have also adopted various hypergraph operations in the mainframe to enhance the spatiotemporal performance. These functional units are out-phase adaptive hypergraphs instead of hard-coded hypergraphs, in-phase hypergraphs, and temporal attention. We find that our in-phase and out-phase hypergraph model (93.88%) carries a greater significance compared to the fixed hypergraph model (92.7%) in terms of accuracy as shown in Table 2. The weighting of each frame according to its contribution to the final output features using temporal attention helps to neglect the insignificant frames and elevates the performance to a maximum extent of 94.15% accuracy.

Table 3. Impact of layer counts and scaling factors in decoder.

Layer Count	Scaling Factor 0.3	0.5	0.7	0.9	1.0
6	93.62	93.48	93.35	92.98	92.40
3	93.69	93.84	93.97	94.15	93.92

Table 4. Effect of input dimension fed to hypergraph generator.

Channel Count	4	6	8	12
Accuracy (%)	93.68	93.90	94.15	93.86

Impact of layer counts and scaling factors in decoder: We apply hybrid learning, combining supervised and unsupervised techniques, to enhance the performance of the proposed AutoregAd-HGformer. A decoder is used on the final classification feature embedding to apply unsupervised learning. As shown in Table 3, we evaluate the optimal layer count and scaling factor for reconstruction error related to the decoder unit. We observe that the three-layered decoder with a scaling factor of 0.9 yields the highest accuracy of 94.15%. We also observe that a smaller scaling

Table 5. Impact of the scaling factor as shown in Fig. 1

.

α	0.1	0.2	0.3	0.4	0.6
Accuracy (%)	93.64	94.15	93.92	93.76	93.67

Table 6. Quantitative comparison of AutoregAd-HGformer on NTU RGB+D 60, NTU RGB+D 120, and NW-UCLA datasets.

Architecture	Method	NTU-60		NTU-120		NW-UCLA
		X-Sub (%)	X-View (%)	X-Sub (%)	X-set (%)	
Graph Convolution	ST-GCN [41]	81.5	88.3	70.7	73.2	-
	2S-AGCN [23]	88.5	95.1	82.5	84.2	-
	Shift-GCN [7]	90.7	96.5	85.9	87.6	94.6
	SGN [42]	89.0	94.5	79.2	81.5	92.5
	CTR-GCN [5]	92.4	96.8	88.9	90.6	96.5
	Info-GCN [8]	93.0	97.1	89.8	91.2	97.0
	HLP-GCN [37]	92.7	96.9	89.0	90.8	96.8
	HD-GCN [14]	93.4	97.2	90.1	91.6	97.2
Graph Transformer	ST-TR [21]	89.9	96.1	81.9	84.1	-
	STST [44]	91.9	96.8	-	-	-
	IIP-Transformer [34]	92.3	96.4	88.4	89.7	-
	FG-STFormer [10]	92.6	96.7	89.0	90.6	97.0
	IGFormer [20]	93.4	96.5	85.4	86.5	-
	MAMP [19]	93.1	97.5	90.0	91.3	-
Hypergraph Convolution	Hyper-GNN [11]	89.5	95.7	-	-	-
	DH-GCN [6]	90.7	96.0	86.0	87.9	-
	Selective-HCN [48]	90.8	96.6	-	-	-
	DST-HCN [36]	92.3	96.8	88.8	90.7	96.6
Hypergraph Transformer	Hyperformer [46]	92.9	96.5	89.9	91.3	96.9
	AutoregAd-HGformer (ours)	**94.15**	**97.83**	**91.02**	**92.42**	**97.98**

factor should be chosen for a larger-layered decoder for optimal performance and vice versa.

Effect of input dimension d fed to hypergraph generator and scaling factor α: The k-means clustering would not perform best for the embedding either with larger dimensions or with smaller dimensions due to "Curse of Dimensionality". As shown in Table 4, we also observe that AutoregAd-HGformer yields the highest performance in terms of accuracy (94.15%) for 8-channel embedding compared to 4, 6, and 12. Similarly, as shown in Table 5, the proposed model yields the highest performance for the scaling factor $\alpha = 0.2$ compared to 0.1, 0.3, 0.4, and 0.6.

5.4. Quantitative Comparison

Table 6 provides a comprehensive comparison of our AutoregAd-HGformer model with state-of-the-art methods across four different architectures: Graph Convolution, Graph Transformer, Hypergraph Convolution, and Hypergraph Transformer. The comparison is based on the top-1 accuracy (%) on three datasets: NTU RGB+D 60, NTU RGB+D 120, and NW-UCLA.

① Graph Convolution Architecture: As shown in Table 6, the recent HD-GCN method is very competitive compared to the proposed transformer-based AutoregAd-HGformer. However, AutoregAd-HGformer achieves a top-1 accuracy of 94.15% and 97.83% on two settings of NTU RGB+D 60, 91.02% and 92.42% on two settings of NTU RGB+D 120, and 97.98% on NW-UCLA datasets, outperforming all other methods based on this architecture.

② Graph Transformer Architecture: As the proposed model adopts the hypergraph transformer technique in its mainframe, we compare its performance to SOTA graph transformer architecture. AutoregAd-HGformer continues to perform exceptionally well in this architecture on each dataset.

③ Hypergraph Convolution Architecture: The conceptualization of the proposed model is centered around hypergraph by grouping the contextually correlated graph nodes in the graph. AutoregAd-HGformer also maintains its superiority with accuracy on the NTU RGB+D 60, NTU RGB+D 120, and NW-UCLA datasets in these settings.

④ Hypergraph Transformer Architecture: As little attention has been paid to hypergraph transformers so far, we found only one paper implemented called Hyperformer. The performance of AutoregAd-HGformer also surpasses the state-of-the-art hypergraph transformer-based architecture by a margin of 0.7% in terms of top-1 accuracy.

Overall, our AutoregAd-HGformer model consistently outperforms state-of-the-art methods across all architectures and datasets, demonstrating its superior performance and effectiveness for action recognition tasks.

5.5. Model-agnostic Performance Comparison

Instead of taking only the Hyperformer [46] as the base encoder, we have also considered DST-HCN [36], Selective-HCN [48], and 3Mformer [32] as the hypergraph encoder and observed the enhancement in model-agnostic performance. As clearly observed in Table 7, all the listed hypergraph encoders with the proposed adaptive hypergraph decoder surpass the solo individual performances.

Table 7. Model-agnostic performance of hypergraph decoder.

Dataset	NTU-60		NTU-120		NW-UCLA
Setting	X-Sub	X-View	X-Sub	X-View	
DST-HCN [36]	90.7	96.0	86.0	87.9	-
DST-HCN*	90.7	95.9	85.9	87.9	94.9
DST-HCN*+Ad HypDec	**91.3**	**96.4**	**86.5**	**88.3**	**95.1**
Selective-HCN [48]	90.8	96.6	-	-	-
Selective-HCN*	90.7	96.6	86.3	88.1	95.0
Selective-HCN*+Ad HypDec	**91.4**	**97.0**	**86.7**	**88.3**	**95.3**
Hyperformer [46]	92.9	96.5	89.9	91.3	96.9
Hyperformer*	92.9	96.4	89.9	91.3	96.8
Hyperformer*+Ad HypDec	**93.65**	**97.25**	**90.49**	**91.91**	**97.43**
3Mformer [32]	94.8	98.7	92.0	93.8	97.8
3Mformer*	94.8	98.6	91.9	93.8	97.8
3Mformer*+Ad HypDec	**95.2**	**98.9**	**92.2**	**94.1**	**98.1**

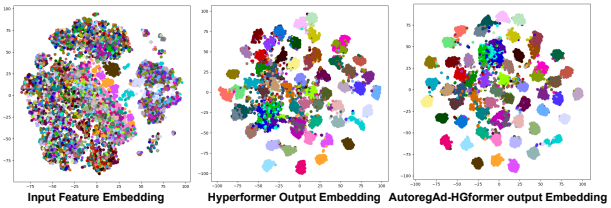

Figure 3. t-SNE [29] of input features (left) and model output feature embeddings.

5.6. Qualitative Comparison

In Fig. 3, we present the t-SNE distribution plot of output embeddings of Hyperformer (middle) and AutoregAd-HGformer (right) trained on the NTU RGB+D 60 dataset. The more organized clustering of skeleton-based action data points in Ad-HFformer highlights its superiority over other hypergraph transformer-based Hyperformer architectures. This figure also indicates that the proposed method can handle skeletal action sequences more effectively by maximizing the intra-class similarity and inter-class separability features, i.e., skeleton-based action sequences from the same class clustering together while those from different classes spread apart.

5.7. Model complexity

The tradeoff between computational complexity and model performance (accuracy in %) is illustrated in Table 8. Our model performs better with fewer parameters than CTR-GCN, Info-GCN, HD-GCN, and DST-CN. Although the proposed AutoregAd-HGformer architecture needs more parameters than ST-GCN and Shift-GCN, it outperforms the above two by a large margin in terms of flops and accuracies. The above ablation is analyzed on the NTU RGB+D 60 dataset.

Table 8. Performance vs Model Complexity.

Models	Publication	Params (M)	Flops (G)	X-Sub/X-View (%)
ST-GCN [41]	AAAI-2018	3.08	16.32	81.5/88.3
Shift-GCN [7]	CVPR-2020	2.76	10.01	90.7/96.5
CTR-GCN [5]	ICCV-2021	5.84	7.88	92.4/96.8
Info-GCN [8]	ICCV-2022	6.28	6.72	92.7/96.9
HD-GCN [14]	ICCV-2023	6.72	6.40	93.0/97.0
DST-HCN [36]	ICME-2023	3.50	2.93	92.3/96.8
Hyperformer [7]	arXiv-2023	2.60	14.8	92.9/96.5
AutoregAd-HGformer	**Proposed**	**3.20**	**15.4**	**94.15/97.83**

6. Misclassification

We observe the semantics of some action sequences in NTU RGB+D 60 datasets are very similar and often misclassified by the present state-of-the-art models. For example: (a) "writing[A12]" as "type on a keyboard[A30]" and vice-versa, (b) "reading[A11]" as "writing[A12]" and vice-versa, (c) "reading[A11]" as "play with the phone/tablet[A29]" and vice-versa, and (d) "writing[A12]" as "play with the phone/tablet[A29]" and vice-versa. We also recognize that using the proposed adaptive hyperedge decoder helps competently reduce the misclassification of the above action sets. Fig. 4 validates the significance of the adaptive hyperedge decoder in the proposed AutoregAd-HGformer.

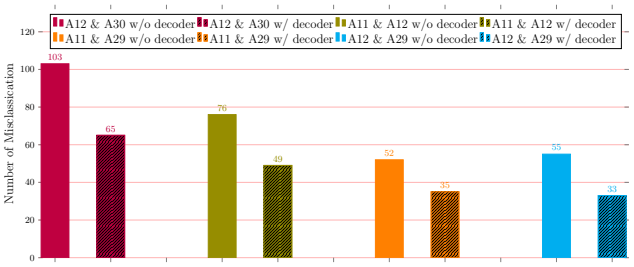

Figure 4. Misclassification between various ambiguous actions (Axx & Axx) of NTU RGB+D 60 dataset before and after implementing adaptive decoder. A11:reading, A12:writing, A29:play with the phone/tablet, A30:type on a keyboard.

7. Conclusion

We propose a novel adaptive hypergraph transformer framework called AutoregAd-HGformer that aggregates multiscale graphs and higher-order contextual semantics with long-range motion features to recognize skeleton sequences for diverse actions. We apply attention mechanisms such as joint-joint self-attention, joint-hyperedge cross-attention, and joint-bone cross-attention to distill frame-level and temporal attention to derive motion-level deep action-dependent features. We also employ channel attention to capture heterogeneous contextual information. The post-extraction decoder helps execute hybrid learning (supervised and self-supervised) and iterative hyperedge clustering to make the model more robust. The novel vector quantized in-phase and model-agnostic out-phase hypergraph generation helps the model to aggregate more robust features for accurate hyperedge calculation. Extensive experiments on multiple datasets show the effectiveness of AutoregAd-HGformer. By incorporating post-extraction adaptive hypergraphs with joint, hyperedge, bone, node, and channel-level self- or cross-attention, AutoregAd-HGformer outperforms the other state-of-the-art architectures on the NTU RGB+D 60, NTU RGB+D 120, and NW-UCLA datasets. The hyperedge-hyperedge self-attention is left for future consideration.

References

[1] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction. In *2021 International Conference on 3D Vision (3DV)*, pages 565–574. IEEE, 2021. 2

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4

[3] Noirrit Kiran Chandra, Antonio Canale, and David B Dunson. Escaping the curse of dimensionality in bayesian model-based clustering. *Journal of machine learning research*, 24(144):1–42, 2023. 4

[4] Xiangyu Chen, Xintao Wang, Jiantao Zhou, and Chao Dong. Activating more pixels in image super-resolution transformer. arxiv 2022. *arXiv preprint arXiv:2205.04437*, 1, 2022. 4

[5] Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13359–13368, 2021. 1, 6, 7, 8

[6] Ke Cheng, Yifan Zhang, Congqi Cao, Lei Shi, Jian Cheng, and Hanqing Lu. Decoupling gcn with dropgraph module for skeleton-based action recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 536–553. Springer, 2020. 7

[7] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 183–192, 2020. 7, 8

[8] Hyung-gun Chi, Myoung Hoon Ha, Seunggeun Chi, Sang Wan Lee, Qixing Huang, and Karthik Ramani. Infogcn: Representation learning for human skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20186–20196, 2022. 1, 7, 8

[9] Sangwoo Cho, Muhammad Maqbool, Fei Liu, and Hassan Foroosh. Self-attention network for skeleton-based human action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 635–644, 2020. 2

[10] Zhimin Gao, Peitao Wang, Pei Lv, Xiaoheng Jiang, Qidong Liu, Pichao Wang, Mingliang Xu, and Wanqing Li. Focal and global spatial-temporal transformer for skeleton-based action recognition. In *Proceedings of the Asian Conference on Computer Vision*, pages 382–398, 2022. 1, 2, 7

[11] Xiaoke Hao, Jie Li, Yingchun Guo, Tao Jiang, and Ming Yu. Hypergraph neural network for skeleton-based action recognition. *IEEE Transactions on Image Processing*, 30:2263–2275, 2021. 1, 2, 3, 7

[12] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019. 12

[13] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. A new representation of skeleton sequences for 3d action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3288–3297, 2017. 1

[14] Jungho Lee, Minhyeok Lee, Dogyoon Lee, and Sangyoun Lee. Hierarchically decomposed graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10444–10453, 2023. 1, 7, 8

[15] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Skeleton-based action recognition with convolutional neural networks. In *2017 IEEE international conference on multimedia & expo workshops (ICMEW)*, pages 597–600. IEEE, 2017. 1

[16] 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. 6

[17] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 816–833. Springer, 2016. 1

[18] Nguyen Huu Bao Long. Step catformer: Spatial-temporal effective body-part cross attention transformer for skeleton-based action recognition. *arXiv preprint arXiv:2312.03288*, 2023. 1, 2

[19] Yunyao Mao, Jiajun Deng, Wengang Zhou, Yao Fang, Wanli Ouyang, and Houqiang Li. Masked motion predictors are strong 3d action representation learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10181–10191, 2023. 1, 3, 7

[20] Yunsheng Pang, Qiuhong Ke, Hossein Rahmani, James Bailey, and Jun Liu. Igformer: Interaction graph transformer for skeleton-based human interaction recognition. In *European Conference on Computer Vision*, pages 605–622. Springer, 2022. 1, 2, 7

[21] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. Spatial temporal transformer network for skeleton-based action recognition. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part III*, pages 694–701. Springer, 2021. 2, 7

[22] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016. 1, 6

[23] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12026–12035, 2019. 1, 7

[24] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Decoupled spatial-temporal attention network for skeleton-based action-gesture recognition. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[25] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Constructing stronger and faster baselines for skeleton-based action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):1474–1488, 2022. 1

[26] Tae Soo Kim and Austin Reiter. Interpretable 3d human action analysis with temporal convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 20–28, 2017. 1

[27] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 1

[28] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 4, 5

[29] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 8, 13

[30] Jiang Wang, Xiaohan Nie, Yin Xia, Ying Wu, and Song-Chun Zhu. Cross-view action modeling, learning and recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2649–2656, 2014. 6

[31] Lei Wang, Du Q Huynh, and Piotr Koniusz. A comparative review of recent kinect-based action recognition algorithms. *IEEE Transactions on Image Processing*, 29:15–28, 2019. 1

[32] Lei Wang and Piotr Koniusz. 3mformer: Multi-order multi-mode transformer for skeletal action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5620–5631, 2023. 7, 8

[33] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015. 1

[34] Qingtian Wang, Shuze Shi, Jiabin He, Jianlin Peng, Tingxi Liu, and Renliang Weng. Iip-transformer: Intra-inter-part transformer for skeleton-based action recognition. In *2023 IEEE International Conference on Big Data (BigData)*, pages 936–945. IEEE, 2023. 7

[35] S. Wang, Y. Zhang, H. Qi, M. Zhao, and Y. Jiang. Dynamic spatial-temporal hypergraph convolutional network for skeleton-based action recognition. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 2147–2152, Los Alamitos, CA, USA, jul 2023. IEEE Computer Society. 1

[36] Shengqin Wang, Yongji Zhang, Hong Qi, Minghao Zhao, and Yu Jiang. Dynamic spatial-temporal hypergraph convolutional network for skeleton-based action recognition. *arXiv preprint arXiv:2302.08689*, 2023. 2, 3, 7, 8

[37] Chao Wei and Zhidong Deng. Accommodating self-attentional heterophily topology into high-and low-pass graph convolutional network for skeleton-based action recognition. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2023. 1, 7

[38] Jinfeng Wei, Yunxin Wang, Mengli Guo, Pei Lv, Xiaoshan Yang, and Mingliang Xu. Dynamic hypergraph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:2112.10570*, 2021. 2, 3

[39] Wenhan Wu, Yilei Hua, Ce Zheng, Shiqian Wu, Chen Chen, and Aidong Lu. Skeletonmae: Spatial-temporal masked autoencoders for self-supervised skeleton action recognition. In *2023 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 224–229. IEEE, 2023. 1, 3

[40] Wentian Xin, Ruyi Liu, Yi Liu, Yu Chen, Wenxin Yu, and Qiguang Miao. Transformer for skeleton-based action recognition: A review of recent advances. *Neurocomputing*, 2023. 1

[41] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1, 7, 8

[42] Pengfei Zhang, Cuiling Lan, Wenjun Zeng, Junliang Xing, Jianru Xue, and Nanning Zheng. Semantics-guided neural networks for efficient skeleton-based human action recognition. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1112–1121, 2020. 6, 7

[43] Songyang Zhang, Xiaoming Liu, and Jun Xiao. On geometric features for skeleton-based action recognition using multilayer lstm networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 148–157. IEEE, 2017. 1

[44] Yuhan Zhang, Bo Wu, Wen Li, Lixin Duan, and Chuang Gan. Stst: Spatial-temporal specialized transformer for skeleton-based action recognition. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3229–3237, 2021. 2, 7

[45] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2914–2923, 2017. 1

[46] Yuxuan Zhou, Zhi-Qi Cheng, Chao Li, Yanwen Fang, Yifeng Geng, Xuansong Xie, and Margret Keuper. Hypergraph transformer for skeleton-based action recognition. *arXiv preprint arXiv:2211.09590*, 2022. 2, 7, 8

[47] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. 1

[48] Yiran Zhu, Guangji Huang, Xing Xu, Yanli Ji, and Fumin Shen. Selective hypergraph convolutional networks for skeleton-based action recognition. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*, pages 518–526, 2022. 1, 3, 7, 8

Autoregressive Adaptive Hypergraph Transformer for Skeleton-based Activity Recognition

-:Supplemetary Material:-

This supplementary material provides additional information about hypergraph convolution, temporal attention in STA-HT block, hypergraph quantizer, comprehensive ablation study, and additional experimental results in sec. 8, 9, 10, 11, and 12 respectively.

8. Hypergraph Convolution

Hypergraph: An undirected weighted graph $G = (V, E, W)$ is a structural representative of non-Euclidean data points in terms of nodes (V) and associated edges (E) along with their weights (W). Each edge in G can only epitomize two nodes. The hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ gives an excellent solution to achieve the semantic relation between more than two nodes. Here, $\mathcal{V}$ represents the nodes, same as in G, $\mathcal{E}$, and $\mathcal{W}$ denote the set of hyperedges and their corresponding weights, respectively. The hypergraph is represented by an incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$

$$\mathbf{H}_{i,j} = \begin{cases} 1 & \text{if node } i \text{ belongs to hyperedge } j \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

A diagonal matrix $\mathbf{H_w} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is used to store the weights of hyperedges. The degree of a node $d_v(.)$ is the sum of the weights of all hyperedges associated with that node v, and the degree of a hyperedge $d_e(.)$ is the number of nodes contained in hyperedge e. $\mathbf{D_v}$ and $\mathbf{D_e}$ are diagonal degree matrices of nodes and hyperedges, respectively.

Hypergraph Convolution: A hypergraph convolution is defined as

$$\mathbf{Y_h} = \mathbf{D_v}^{-1/2}\mathbf{H}\mathbf{H_w}\mathbf{D_e}^{-1}\mathbf{H^T}\mathbf{D_v}^{-1/2}\mathbf{X_1}\theta \tag{22}$$

where $\mathbf{X_1} \in \mathbb{R}^{|\mathcal{V}|\times C_1}$ is the input node feature matrix and $\theta \in \mathbb{R}^{C_1\times C_2}$ is the convolutional filter on hypergraph. We use the notation $HGConv(X, \mathbf{H^n}, \mathbf{H_w}^{\mathbf{n}})$ to represent hypergraph convolution on the input feature X.

9. Temporal attention in STA-HT block

In the STA-HT block, temporal attention is applied to the hyperedge features H_f. Attention weights for different timestamps are calculated and multiplied with H_f. We adopt a squeeze-excitation block to calculate the attention weights as in [12]. A linear layer squeezes the number of temporal dimensions with ReLU activation, and another linear layer excites the temporal dimension to bring it back to its original count with sigmoid activation. These weights are multiplied to residual hyperedge features H_f to get attentive features.

10. Hypergraph Quantizer

A codebook, which is formed by the trainable vectors $\{c_1, ..., c_k\}$, is utilized to group the nodes that form input-dependent hyperedges. All these vectors can be represented by a tensor $C \in \mathbb{R}^{k\times D}$. The intermediate output embedding $p \in \mathbb{R}^{V\times d}$ for a single human skeleton from the second FAHT unit is passed through a decoder which is similar to the HypDec block in terms of architecture but comprising a different set of parameters. The j^{th} node is assigned to hyperedge ind_j in the following manner.

$$ind_j = argmin_i \|c_i - p_j\|_2 \tag{23}$$

Subsequently, we pass the incidence matrix Hq^n, representing the input-dependent hyperedge, to the next FAHT block. We optimize the distance between vector priors and node embeddings by allowing the vectors to be learnable, thereby enhancing adaptability. This approach closely resembles a clustering process. The mean squared error between the node embedding and c_j is assigned as one of the loss functions for the model training.

$$\mathcal{L}_{\text{quant}} = (1/V)\sum_{i=1}^{V} \|c_{ind_i} - sg(p_i)\|_2^2. \tag{24}$$

Here, sg denotes the gradient operator with respect to p_i, which is set as zero during backpropagation.

To learn the weight of hyperedges, we use quantized embeddings, which are defined as

$$p_{quant_j} = c_{ind_j}. \tag{25}$$

We calculate ${Hq_w}^n$ after determining the weights of each hyperedge by advancing the quantized node embeddings through an MLP layer. Finally, ${Hq_w}^n$ is passed to the next set of FAHT blocks.

The p_{quant_j} generation during the quantization process is a non-differentiable phenomenon, which is bypassed during backpropagation. We follow the straight-through estimator approach [refer to paper] for backpropagation which means the gradients from p_{quant} are directly copied to p.

11. Comprehensive Ablation Study

We evaluate our model using different values of related hyperparameters to get the best performance. These hyperparameters are the number of channels in transformer layers (c), the number of transformer blocks (L), and the number of hyperedges (k). Fig. 5 shows the plots for model performance (accuracy in %) vs number of epochs for the NTU RGB+D 60 dataset in the X-sub setting. As shown in this figure, we obtained the best results for $c = 216$, $L = 10$, and $k = 5$. We also generate the t-SNE plots of the above hyperparameters, taking the same corresponding values as shown in Fig. 6 to consolidate our ablation. The best separation of different classes is visually evident for 10 transformer blocks with a channel count of 216 and hyperedges of 5.

12. Experimental Results

Apart from the above, we have also analyzed the impact of various modules on individual class performances. As shown in Table 9, we have seen that the accuracies related to most of the individual classes are increasing due to the effectiveness of various modules in AutoregAd-HGformer.

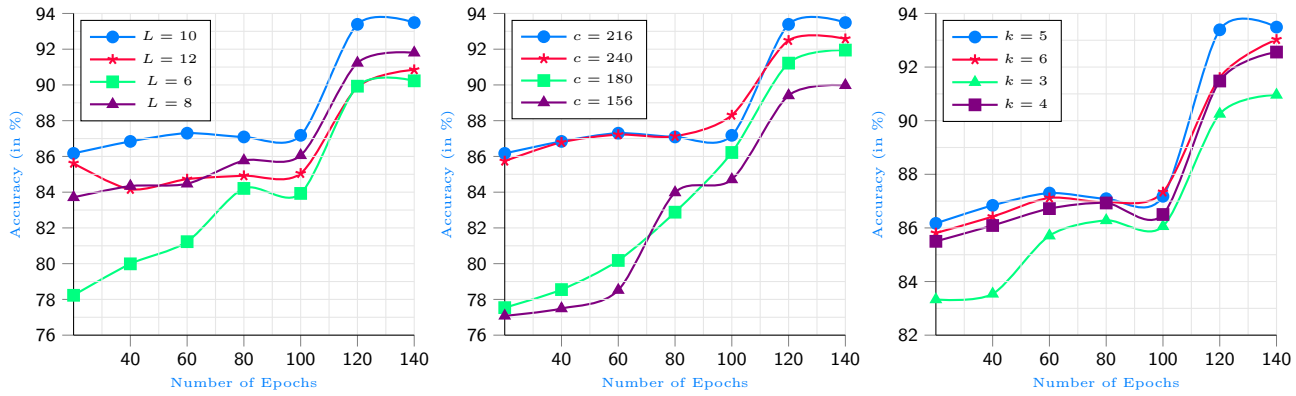

Figure 5. Epoch-wise performance (accuracy in %) comparison of the proposed AutoregAd-HGformer for **Left:** transformer block counts (L), **Middle:** transformer channel counts (c), **Right:** hyperedge count (k). [on NTU RGB+D 60(X-sub)]

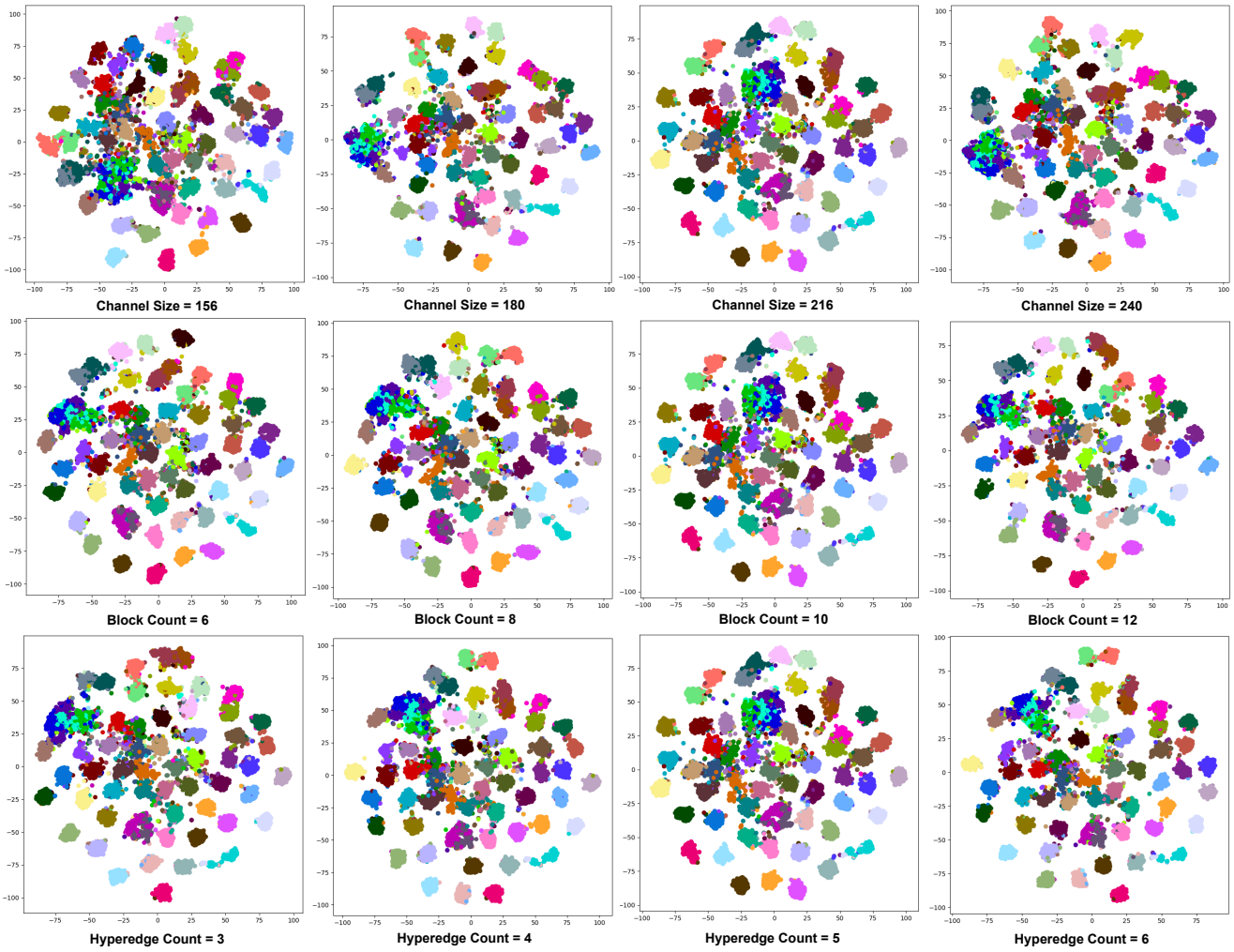

Figure 6. t-SNE [29] plot of **Top:** transformer block counts (L), **Middle:** transformer channel counts (c), **Bottom:** hyperedge count (k). [on NTU RGB+D 60(X-sub)]

Table 9. Class-wise performance related to the impact of various modules in AutoregAd-HGformer on NTU RGB+D 60 X-Sub setting. OHG: Out-phase Hypergraph Generator, RL: Reconstruction Loss, THA: Temporal Hypergraph Attention, IHQ: In-phase Hyergraph Quantizer. The enhanced and reduced class-wise accuracy are given by ↑ and ↓ respectively.

Methods	**Baseline**	**OHG**	**OHG+RL**	**OHG+RL+THA**	**OHG+RL+THA+IHQ**
Params(M) / **Action Labels**	2.60	2.75	2.95	3.10	3.20
drink water	87.70	88.50 (0.8↑)	88.52	88.25	88.95
eat meal/snack	78.89	79.40(0.51↑)	78.78	80.85	81.35
brushing teeth	91.68	92.50(0.82↑)	92.20	91.94	91.86
brushing hair	92.44	92.92(0.48↑)	93.95	95.12	94.26
drop	93.66	93.94(0.28↑)	94.85	95.56	94.78
pickup	98.38	98.24(0.11↓)	98.55	97.45	98.45
throw	94.48	94.22(0.26↓)	94.95	96.10	94.25
sitting down	98.88	99.25(0.37↑)	99.10	97.97	99.13
standing up (from sitting position	98.91	99.24(0.33↑)	98.90	99.15	99.27
clapping	86.86	86.80(0.06↓)	86.85	88.10	87.10
reading	61.79	67.81(6.02↑)	67.92	66.98	67.18
writing	71.63	77.75(6.12↑)	76.85	78.00	77.32
tear up paper	96.49	96.63(0.14↑)	96.60	96.06	96.57
wear jacket	98.58	98.71(0.13↑)	98.55	97.43	98.55
take off jacket	98.95	99.17(0.22↑)	99.25	99.48	99.18
wear a shoe	67.88	70.01(2.13↑)	71.74	73.16	73.78
take off a shoe	85.88	86.10(0.22↑)	86.37	85.00	84.00
wear on glasses	95.25	94.80(0.45↓)	93.67	95.95	95.15
take off glasses	96.46	96.47(0.01↑)	97.56	98.58	95.62
put on a hat/cap	98.42	99.00(0.58↑)	98.65	98.75	98.75
take off a hat/cap	98.95	99.18(0.23↑)	98.32	97.95	98.95
cheer up	94.83	94.90(0.07↑)	95.20	95.89	94.89
hand waving	96.26	96.00(0.26↓)	97.10	97.25	96.95
kicking something	97.76	97.92(0.16↑)	97.68	97.85	98.35
reach into pocket	87.55	87.64(0.09↑)	88.37	87.97	91.57
hopping (one foot jumping)	98.94	99.26(0.32↑)	99.11	98.91	98.79
jump up	99.25	99.60(0.35↑)	99.45	98.75	99.25
make a phone call/answer phone	93.13	93.55(0.42↑)	94.35	95.55	96.53
playing with phone/tablet	79.82	83.60(3.78↑)	81.38	81.47	82.56
typing on a keyboard	76.58	81.17(4.59↑)	83.27	84.66	84.86
pointing to something with finger	82.88	83.24(0.36↑)	83.95	84.34	85.49
taking a selfie	95.42	95.80(0.38↑)	96.06	96.66	97.82
check time (from watch)	94.32	94.91(0.59↑)	94.95	95.40	94.67
rub two hands together	92.91	92.70(0.21↓)	93.15	92.87	93.96
nod head/bow	98.85	99.35(0.50↑)	99.48	98.18	98.83
shake head	96.48	97.60(1.12↑)	98.35	99.02	98.61
wipe face	88.39	89.70(1.31↑)	89.89	90.39	91.76
salute	96.38	96.30(0.08↓)	96.47	97.05	96.89
put the palms together	98.86	98.59(0.27↓)	98.46	98.83	98.96
cross hands in front (say stop)	97.72	98.35(0.63↑)	99.15	98.15	98.94
sneeze/cough	85.24	84.50(0.74↓)	84.75	85.64	86.62
staggering	99.48	99.63(0.15↑)	99.48	98.88	99.26
falling	99.52	99.87(0.35↑)	99.82	99.48	99.36
touch head (headache)	88.98	89.98(1.00↑)	90.10	91.55	91.87
touch chest (stomachache/heart pain)	96.68	96.35(0.33↓)	96.53	96.35	97.78
touch back (backache)	97.49	97.71(0.22↑)	97.48	97.03	98.72
touch neck (neckache)	92.57	92.22(0.35↓)	93.67	93.25	95.44
nausea or vomiting condition	89.08	89.40(0.32↑)	89.22	89.87	90.74
use a fan/feeling warm	92.46	92.02(0.44↓)	93.19	93.18	94.26
punching/slapping other person	95.23	95.99(0.76↑)	95.72	94.99	95.63
kicking other person	97.37	97.88(0.51↑)	97.83	97.68	98.93
pushing other person	99.15	99.60(0.45↑)	99.37	98.83	99.42
pat on back of other person	95.71	95.71(0.00↑)	96.36	96.68	96.88
point finger at the other person	94.80	94.92(0.12↑)	96.00	96.45	96.18
hugging other person	99.55	99.60(0.05↑)	99.55	99.07	99.28
giving something to other person	97.55	97.47(0.08↓)	96.72	97.83	97.71
touch other person's+pocket	98.34	98.90(0.56↑)	97.62	98.75	98.91
handshaking	98.27	98.63(0.36↑)	98.78	97.36	99.07
walking towards each other	99.69	99.66(0.03↓)	99.54	98.86	99.41
walking apart from each other	98.35	99.13(0.78↑)	99.32	98.91	99.49
Overall	**92.90**	**93.50**(0.60↑)	**93.65**	**93.79**	**94.15**