STr-GCN: Dual Spatial Graph Convolutional Network and Transformer Graph Encoder for 3D Hand Gesture Recognition

Rim Slama, Wael Rabeh, Hazem Wannous

▶ **To cite this version:**

Rim Slama, Wael Rabeh, Hazem Wannous. STr-GCN: Dual Spatial Graph Convolutional Network and Transformer Graph Encoder for 3D Hand Gesture Recognition. IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG 2023), Jan 2023, Waikoloa Beach, HI, United States. 10.1109/FG57933.2023.10042643 . hal-04711580

HAL Id: hal-04711580
https://hal.science/hal-04711580v1

Submitted on 2 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

STr-GCN: Dual Spatial Graph Convolutional Network and Transformer Graph Encoder for 3D Hand Gesture Recognition

Rim Slama[1], Wael Rabah[1] and Hazem Wannous[2]
[1] LINEACT Laboratory, CESI Lyon, France
[2] IMT Nord Europe, CRIStAL UMR CNRS 9189, France

***Abstract*— Skeleton-based hand gesture recognition is a challenging task that sparked a lot of attention in recent years, especially with the rise of Graph Neural Networks. In this paper, we propose a new deep learning architecture for hand gesture recognition using 3D hand skeleton data and we call STr-GCN. It decouples the spatial and temporal learning of the gesture by leveraging Graph Convolutional Networks (GCN) and Transformers. The key idea is to combine two powerful networks: a Spatial Graph Convolutional Network unit that understands intra-frame interactions to extract powerful features from different hand joints and a Transformer Graph Encoder which is based on a Temporal Self-Attention module to incorporate inter-frame correlations. We evaluate the performance of our method on three benchmarks: the SHREC'17 Track dataset, Briareo dataset and the First Person Hand Action dataset. The experiments show the efficiency of our approach, which achieves or outperforms the state of the art. The code to reproduce our results is available in this link.**

Keywords Hand gesture recognition, Skeleton, Graphs, Graph Convolutional Networks, Attention, Transformers

I. INTRODUCTION

Hand gesture recognition has recently attracted great attention in the human-machine interaction field thanks to the opportunities it offers in different contexts such as healthcare, industry and entertainment. Based on the chosen modality, hand gesture recognition approaches are split into two categories: image-based approaches which employ RGB or RGB-D image sequences and skeleton-based methods which use sequences of 2D or 3D euclidean coordinates of the hand joints. The rapid development of low-cost depth sensors such as Microsoft Kinect and Intel RealSense, coupled with quick advances in hand pose estimation research, has enabled to record RGB, RGB-D and hand skeleton data with high precision.

Skeleton data provides efficient computation and storage, and it has shown its effectiveness and robustness against noise [1]. However, it lies in a non-euclidean space and needs to be considered in an adequate geometric framework. Recently, efficient representation of the skeleton data is widespread using the natural graph representation of the hand skeleton. In particular, using the graph representation leads to the most successful representative work in the skeleton-based action recognition ST-GCN [2]. This latter exploits joint connections using graph convolutions in both spatial and temporal domains. However, the self-attention mechanism has recently demonstrated its capacity to extract powerful temporal information [3]. Motivated by that, we propose a graph based approach that attempts to capture the spatial dependencies among the joints using a Spatial Graph Convolutional Network (S-GCN) and the temporal correlation using a proposed Transformer Graph Encoder (TGE) which takes advantage of a self-attention module. Coupling these two proposed modules has shown the effectiveness of our approach on hand gesture recognition applications. Three benchmarks were tested for validation: SHREC'17 Track dataset [4], Briareo dataset [5] and FPHA dataset [6].
Our contributions can be summarized as follows:

- We introduced a temporal transformer graph based module, which extracts correlations locally between pairs of nodes, taking advantage of the hand graph structure.
- We studied different data preprocessing operations on 3D-skeleton sequences and their influence on the recognition performance.
- Competitive results are achieved on main skeleton-based hand gesture benchmarks.

The rest of this paper is structured as follows. In Section II, related works on GCN, Self-Attention and transformer based hand gesture recognition approaches are reviewed. In Section III, our proposed approach is described. In Section IV, we report our experimental evaluations before concluding in Section V.

II. RELATED WORK

Skeleton-based hand gesture recognition has become an active research area in recent years, and it has been studied extensively, especially with the rise of deep learning. This led to the development of many advanced skeleton-based approaches [7], [8], [9], [10], [11], [12], [13], [14], [15].In this work, we only focus on recent hand gesture and action recognition related works that are based on Graph Convolutional Networks(GCNs) [16], Self-Attention and Transformers [17].

One of the first approaches that use GCNs on skeleton data was ST-GCN, proposed by Yan et al. [2], in which they construct a spatio-temporal graph from a 3D skeleton body. Spatial graph convolution and temporal convolution layers were introduced to extract the adequate features from the body graph sequence. Some other approaches developed new architectures inspired by ST-GCN. AS-GCN [18] introduced new modules to the ST-GCN architecture that capture actional and structural relationships. This helps to overcome their disregard for hidden action-specific joint

979-8-3503-4544-5/23/$31.00 ©2023 IEEE

correlations. Non-local graph convolutions [19] proposed to learn a unique individual graph for each sequence. Focusing on all joints, they decide whether there should be connections between pairs of joints or not. 2S-AGCN [20] built a 2 stream architecture to model both the skeleton data and second-order information such as the direction and length of the bones. They used Adaptive GCN (AGCN) [21], which learns 2 adjacency matrices individually for each sequence and uniformly shared between all the sequences. The same authors later proposed MS-AAGCN [22] that improves on their previous architecture [21] by modeling a third stream called the motion stream. AAGCN was proposed, which further enhances on AGCN with a spatio-temporal attention module, enabling the learned model to pay more attention to important joints, frames and features. Transformers are sequence models introduced primarily in NLP, which perform better feature extraction than recurrent models thanks to the self-attention mechanism. The most recent and notable related works include STA-GCN [3] which used spatial and temporal self-attention modules to learn trainable adjacency matrices. In STA-RES-TCN [11], spatio-temporal attention was used to enhance residual temporal convolutional networks. The use of the attention mechanism enables the network to concentrate on the important frames and features and eliminate the unimportant ones that frequently add extra noise. DG-STA [23] proposed to leverage the attention mechanism to construct dynamic temporal and spatial graphs by automatically learning the node features and edges. ST-TR [24] proposed a Spatial and temporal Self-Attention modules used to understand intra-frame interactions between different body parts and interpret hidden inter-frame correlations. In this work, in order to analyze the local and global relationships between the hand joints in both spatial and temporal domains, we adopt a fully graph based approach that exploits this non-linear structure of the hand in these domains.

III. Our Approach

In our method, we take advantage of two models: a Spatial Graph Convolutional Network (S-GCN) is used for spatial information extraction from graphs, coupled with a Transformer Graph Encoder (TGE) for capturing temporal features in sequences. Fig. 1 describes different units of our proposed architecture for skeleton-based hand gesture recognition.

Having a sequence of 3D hand skeletons, we use an adjacency matrix to construct a graph sequence. First, in the spatial domain, S-GCN is used to extract hand features at each frame taking advantage of the natural graph structure of the hand skeleton. Then, in a temporal domain and respecting the graph structure of the spatial features, a transformer graph encoder is proposed to extract inter-frame relevant features. Finally, a global pooling operation is used to aggregate the graph into a representation that can be interpreted by our classifier.

A. Formulation

The proposed gesture recognition approach can be defined as a function denoted by GT:

$$\begin{aligned} &GT : R^{\gamma * \lambda * \phi} \rightarrow R^{C} \\ &GT \mapsto Y(T(G(S) + PE)) \end{aligned} \tag{1}$$

This function predicts a probability distribution over C gesture classes from a sequence $S \in R^{\gamma * \lambda * \phi}$ sampled out of the input set of 3D skeleton sequences. We denote, by γ the sequence length, λ the number of hand skeleton nodes constructing the graph and ϕ the number of features per node.

G is the function of the S-GCN module, PE stands for the positional encoding and T represents the TGE module and Y represents the classifier. GT can be decomposed into three operations:

The first operation G corresponds to the **S-GCN** for spatial features extraction from a hand skeleton:

$$\begin{aligned} &G : R^{\gamma * \lambda * \phi} \rightarrow R^{\gamma * \lambda * d_{\text{model}}} \\ &\hat{g} = G(S) \end{aligned} \tag{2}$$

d_{model} is the embedding size of each graph node. An embedding is a numerical vector representation of a complex structured data object. It is calculated such that 2 similar objects would have 2 similar embedding vectors. In our case, $\hat{g}$ is a collection of embedding vectors of each node of the graph.

The second operation T corresponds to the TGE module for temporal feature extraction from inter-frame hand joints:

$$\begin{aligned} &T : R^{\gamma * \lambda * d_{\text{model}}} \rightarrow R^{\gamma * \lambda * d_{\text{model}}} \\ &\hat{t} = T(\hat{g} + PE) \end{aligned} \tag{3}$$

PE is the Positional Encoding vector. The sequence in its current state doesn't carry information about the position of each frame in the sequence. Thus, we add a PE vector to $\hat{g}$, which is a vector that encodes the position of each frame in the sequence. You can refer to Vaswani et al. [17] for more information about the PE operation.

The third operation Y is the classifier. First, we apply a global pooling operation that transforms a sequence of temporal graph features into an aggregated feature map $\hat{f} \in R^{d_{\text{model}}}$. Then a FC layer maps the aggregated temporal features into the C potential gesture classes respecting the following formula:

$$\begin{aligned} &Y : R^{d_{\text{model}}} \rightarrow R^{C} \\ &\hat{y} = Y(\hat{f}) \end{aligned} \tag{4}$$

We denote by $\hat{y}$ the probability distribution over C gesture classes.

B. Spatial Graph Convolutional Networks (S-GCN)

The spatial graph convolution operation is a weighted average aggregating the features of each node with those of all neighboring nodes to produce a new feature vector for the former. This vector contains information about the current node, its neighbors in the same frame, and the importance degree of the connection between them in the hand.

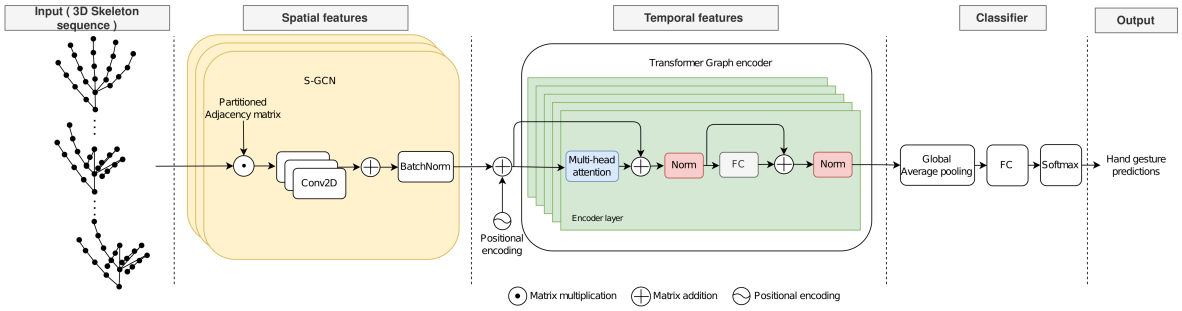


Fig. 1: Overview of our approach.

In order to construct our spatial graph, we need to build an adjacency matrix that represents the connections between different joints in the hand skeleton. We formulate the final adjacency matrix $\mathrm{A_k}$ as follows:

$$A_{\mathrm{k}} = {D_{\mathrm{k}}}^{-1/2}.(\tilde{A}_{\mathrm{k}} + I).{D_{\mathrm{k}}}^{-1/2}, D_{\mathrm{ij}} = \sum_{k}^{K_{\mathrm{a}}}({\tilde{A}_{\mathrm{k}}}^{ij} + I_{\mathrm{ij}}) \quad (5)$$

$\tilde{A}_{\mathrm{k}}$ is the adjacency matrix of the fixed undirected graph representing the connections between the hand joints. If we apply the graph convolution on $\tilde{A}_{\mathrm{k}}$, the result will not contain the features of the node itself. We add I, an identity matrix, to represent the self-connections of the nodes. As $(\tilde{A}_{\mathrm{k}}+I)$ is a binary and not normalized matrix. We multiply it by ${D_{\mathrm{k}}}^{-1/2}$ (the inverse of the degree matrix of the graph) on both sides to normalize it. Then, we use the following formula to compute the graph convolution:

$$G(S) = \sum_{k}^{K_{\mathrm{a}}}(S.A_{\mathrm{k}}).W_{\mathrm{k}}, \;\; S \in R^{\gamma * \lambda * \phi} \quad (6)$$

Where S is a 3D skeleton sequence and K_{a} is the kernel size on the spatial dimension, which also matches the number of adjacency matrices. The number of adjacency matrices depends on the used partitioning strategy, which we will explain below. W_{k} is a trainable weight matrix and is shared between all graphs to capture common properties.
For each kernel, $S.A_{\mathrm{k}}$ calculates the weighted average of the features of each node with its neighboring nodes, which is then multiplied by the weights' matrix $(S.A_{\mathrm{k}}).W_{\mathrm{k}}$. The features calculated by all the kernels are then summed up to form one feature vector per node. In this module, we are inspired by the graph convolution from ST-GCN [2], which uses a similar Graph Convolution formulation to the one proposed by Kipf et al. [16]. We use the partitioning and edge importance techniques to extract more informative features about the node's neighbors and the edges connecting the nodes.

1) Partitioning: We adopt the partitioning technique introduced by Yan et al. [2]. In their paper, they suggested three partitioning strategies: uni-labeling, distance partitioning and spatial configuration partitioning. We have decided to work with the distance partitioning, which we adapt to the 3D hand skeleton graph (see Fig. 2). We chose this strategy because it allows us to capture both local and global features of the hand skeleton. This strategy consists in setting the partitions

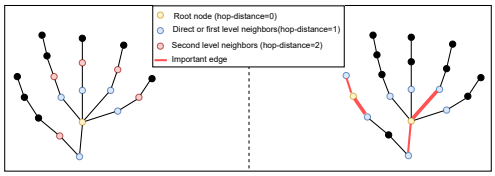


Fig. 2: The hand on the left shows the partitioning using the "distance" strategy". The one on the right shows the impact of edge importance. The bolder the edge, the more it contributes to the movement of the root node.

according to the node's hop distance in comparison to the root node. The hop distance is the number of edges that separate two nodes.

2) Edge importance: This operation is useful when a node contributes to the motion of several surrounding nodes, but these contributions are not equally significant. This mechanism adds a learnable mask for each convolution layer that learns the contribution of each edge to the movement of different parts of the hand. Then, node features are scaled according to the contribution of their edges to their neighboring nodes.

C. Transformer Graph Encoder (TGE)

We propose a Transformer Graph Encoder (TGE) module, which learns the inter-frame correlations locally between joints. These local features are then used to learn the global motion of the hand. As shown in Fig. 1, this module is composed of multiple identical encoder layers stacked together, with each one feeding its output to the next. Each encoder layer applies two operations: **multi-head attention** and a simple **fully connected feed-forward network**. Following He et al. [25] and Ba et al. [26], we respectively employ a skip connection around each of the two operations followed by a layer normalization step. This operation improves deeper models, making them easier and faster to optimize without sacrificing their performance.

This TGE module was inspired from the encoder block introduced in the transformer paper [17]. The original Transformer Encoder (TE) handles sequences of words, which are represented as numerical vectors. However, our hand's spatial

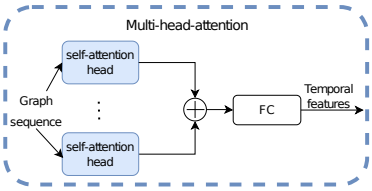


Fig. 3: Multi-head self-attention mechanism.

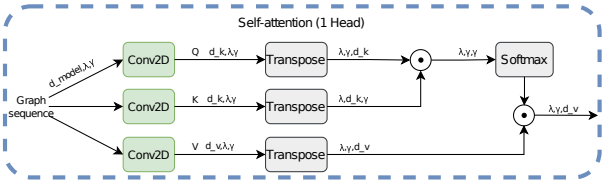


Fig. 4: proposed self-Attention head.

features are represented by a graph. Thus, we have to adapt 'TE' to work on a non-linear graph structure. We alter the self-attention mechanism so that it is computed between the individual nodes of a graph sequence.

Multi-head attention: Multi-Head Attention is a self-attention layer that models the inter-frame relationship between the hand joints. An example of the used multi-head attention module is represented in Fig. 3. To formulate it in the following, we set the number of heads to four:

$$mhAtt(x) = FC(Att1 \bigoplus Att_2 \bigoplus Att_3 \bigoplus Att_4) \quad (7)$$

$\bigoplus$ is the concatenation operator. Att_{i} is a self-attention head. Each head is initialized differently. In theory, this should allow each head to extract different features from the graph sequence. FC is a fully connected layer that projects the concatenation of our attention heads into a 512-dimensional feature space. Attention is computed between each joint $i \in [1..\lambda]$ at frame t, and its corresponding joint in all other frames $t \in [1..\gamma]$. Self-attention Att_{i} is denoted by this formula:

$$\begin{aligned} &Att_{\mathrm{i}} : R^{\gamma*\lambda*d_{\mathrm{v}}} \rightarrow R^{\gamma*\lambda*d_{\mathrm{v}}} \\ &Att_{\mathrm{i}}(x) = softmax(Q_{\mathrm{i}}.K_{\mathrm{i}}/\sqrt{d_{\mathrm{k}}}).V_{\mathrm{i}} \end{aligned} \quad (8)$$

where $x = \hat{g} + PE$ is the spatial features, $\mathrm{W_{Qi}}$, $\mathrm{W_{Ki}}$ and $\mathrm{W_{Vi}}$ are trainable weight matrices. $\mathrm{Q_i}$, $\mathrm{K_i}$ and $\mathrm{V_i}$ are independent projection matrices of the joints spatial feature vectors and are computed as follows: $Q_{\mathrm{i}} = xW_{\mathrm{Qi}} \in R^{\lambda*\gamma*d_{\mathrm{k}}}$, $K_{\mathrm{i}} = xW_{\mathrm{Ki}} \in R^{\lambda*\gamma*d_{\mathrm{k}}}$, $V_{\mathrm{i}} = xW_{\mathrm{Vi}} \in R^{\lambda*\gamma*d_{\mathrm{v}}}$. $d_{\mathrm{v}} = 32$ is the feature size of the matrix V_{i}. $d_{\mathrm{k}} = 32$ is the feature size of matrices K_{i} and Q_{i}. It is also used as a scaling factor, $Q_{\mathrm{i}}.K_{\mathrm{i}}$ is multiplied by $1/\sqrt{d_{\mathrm{k}}}$ to scale large dot product values.

The operation in equation 8 is called the scaled dot product, which is calculated locally between the joints. An example of the self-attention head is shown in Fig. 4.

IV. Experiments and Results

In this section, we provide details about the datasets and training protocols that we use for our experiments. We discuss our training details and interpret our ablation study of our approach. Finally, a comparison of our results with the state-of-the-art approaches is performed on three datasets.

A. Datasets

SHREC'17 TRACK [4]: It is one of the first benchmark for the recognition of hand gestures recorded in 2 configurations: 14 gestures performed using one finger and 28 gestures performed with the full hand. Each gesture is performed between 1 and 10 times by 28 participants totaling 2800 sequences. In the experiments, this data is divided into a 70/30 split.

Briareo [5]: Recently released in 2019, this dataset was mainly collected in the automotive context for applications that aim to reduce driver inattention. The dataset contains 12 gestures performed by 40 different subjects with the right hand, where each gesture is repeated three times. The entire dataset contains 1440 sequences with subjects from 1 to 26 used for training, 27 to 32 used for validation and 33 to 40 used for testing.

FPHA [6]: The First Person Hand Action dataset provides dynamic hand action sequences of subjects performing daily life tasks. In total 1175 samples, containing 6 actors 45 different gestures performed by manipulating 26 objects in 3 scenarios. We use the 1:1 data split ratio proposed in the original paper, with 600 action sequences used for training and 575 for testing.

B. Training details

We conduct our experiments using PyTorch on an NVIDIA Quatro RTX 6000. Adam was used as an optimizer and Cross-entropy as the loss function. For our hyperparameters, a batch size of 32 was chosen for training. The initial learning rate was set to 1e-3, reduced by a factor of 2 if the learning stagnates and the loss doesn't improve in the next 5 epochs. The training stops if the validation accuracy doesn't improve in the next 25 epochs. We set the number of encoders to 6, the number of heads for the multi-head attention to 8, the value of d_{model} to 128 and we used the mish activation function through all of our layers. Taking into account the context of each, a fixed number of frames γ was chosen for each dataset: 30 for SHREC'17, 40 for Briareo and 100 for FPHA dataset. To avoid overfitting and improve our model performance, we choose to augment data by random moving. Introduced in Yan et al.[2], it is a form of random affine transformations applied to the sequence to generate an effect similar to moving the angle of the view point of a camera on playback. We also augment data by noise augmentation, which consists in adding very small values of random noise to the input sequence. This also prevents overfitting and allows the model to generalize better. L1 [27] and L2 [28] Regularization techniques and dropout with a rate of 0.3 are used. We initialize the weights of our model by the Xavier initialization [29] such that the variance of the activation functions is the same across every layer. This initialization contributes to a smoother optimization and prevents exploding or vanishing gradients.

C. Ablation study

We explore how different operations on our architecture and data can affect the performance of our approach. All results in this study are carried out on the Briareo dataset (see Table I).

Comparing the original graph convolution operation extracted from [16] and our proposed S-GCN (inspired by st-gcn [2]), we prove the benefits of using such module for spatial features extraction. This module uses multiple kernels and edge importance, allowing the extraction of richer features from the graphs nodes and edges.

In order to set a partitioning strategy that helps to design an informative label map, we report results on three different partitioning strategies. The distance strategy achieves the best results because it gives the best low-level formulation of the spatial features representing the hand joints.

Method	Briareo(%)
GCN+TGE	78.31
S-GCN+TGE	83.34
S-GCN+Partitioning(Uni-labeling)+TGE	72.58
S-GCN+partitioning (Spatial Configuration)+TGE	92.19
S-GCN+Partitioning (distance)+TGE	95.92
S-GCN+Partitioning (distance)+Edge importance+TGE	96.64

TABLE I: Recognition accuracy (%) on Briareo dataset considering different model configurations.

The uni-labeling strategy provides limited information about the graph adjacency. The spatial configuration strategy provides information about the joints in relation to the barycenter of the hand, and this is not very informative in the case of the hand skeleton. The combination of S-GCN, distance partitioning and edge importance performed the best. This shows the capacity of the distance partitioning strategy and edge importance in extracting local and global details from the 3D skeleton sequence. We will retain the best setup in the following experiments.

D. Comparison with state-of-the-art approaches

We evaluate our method by comparing it to other hand gesture recognition approaches that use the 3D-skeleton modality. We conduct this comparison on 3 datasets: SHREC'17 Track, Briareo and FPHA. We collect most of the results in Table II from [12]. However, due to the lack of experiments on Briareo and FPHA, we train ST-GCN [2] and DG-STA [23] on these 2 datasets. For the fairest comparison, we use the code that they publicly released, and we train their models using the same protocol. For our first experiments, We test our methods' ability in distinguishing specific details and patterns of an action out of numerous actions. To achieve this, we experiment on FPHA dataset, which contains up to 45 different classes of actions. The results in Table II show that our method achieves competitive results, despite that it wasn't designed to interpret objects and actions that include an interaction with objects.

For SHREC'17 dataset [4], our method outperforms all approaches that don't exploit the graph representation of the hand. This demonstrates that understanding the local relationships between the joints, and detecting their inter-frame correlations, are decisive operations.

Method	FPHA(%)	SHREC'17(%)		Briareo(%)
		14G(%)	28G(%)	
Huang et al.[14]	84.35	-	-	-
Huang et al.[15]	77.57	-	-	-
Caputo et al.[8]	-	89.5	-	-
SoCJ+HoHD+HoWR[9]	-	88.2	81.9	-
SEM-MEM+WAL[10]	-	90.83	85.95	-
Res-TCN[11]	-	91.1	87.3	-
STA-Res-TCN[11]	-	93.6	90.7	-
HPEV+HMM+FRPV[12]	90.96	92.5	88.8	-
ST-TS-HGR-NET[13]	**93.22**	94.29	89.4	-
ST-GCN[2]	85.16	92.7	87.7	93.73
DG-STA[23]	78.51	**94.4**	**90.7**	90.91
Ours	91.16	93.39	89.20	**96.64**

TABLE II: Recognition accuracy (%) of our method in comparison with other state-of-the-art approaches.

On Briareo dataset, where the gestures are performed by 40 subjects, we test the ability of our method to recognize gestures independently of the subject. We achieve a 96.64% accuracy, outperforming other state-of-the-art methods on this benchmark. This demonstrates the intra-class robustness of our method and its capacity to recognize the action while ignoring subject related features. The confusion matrix of our approach in Fig. 5a compared to that of st-gcn in Fig. 5b shows that our method achieves similar or better performance on most of the gesture classes. Our method outperforms st-gcn, especially with gestures that are executed in opposite directions. We conclude that this is due to self-attention being computed between the individual joints of all the frames, allowing the extraction of more details, compared to Temporal Convolutional Networks used in st-gcn that perform temporal features extraction on the whole sequence at once. Through our experiments, we show that our method performs well on multiple benchmarks with different characteristics, proving its robustness and stability.

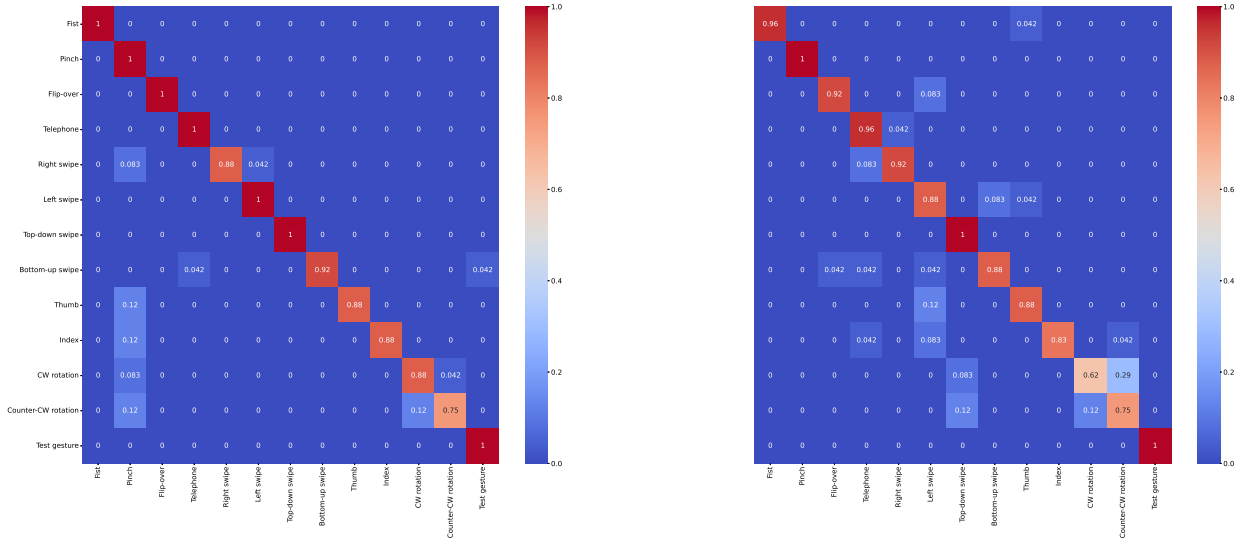
V. Conclusion

In this paper, we proposed a new deep learning architecture STr-GCN for 3D skeleton-based hand gesture recognition. It utilizes a Spatial Graph Convolutional Network in the spatial domain and a Transformer Graph Encoder in the temporal domain to extract informative features from a graph sequence. We performed extensive experiments and have demonstrated the robustness of our method in dealing with multiple datasets that have different characteristics and taken in different contexts. In future works, it could be interesting to consider a multi-modal approach coupled with the one proposed in order to exploit RGB, depth and IR modalities. We intend to evolve our architecture for the online hand gesture recognition task required in many Human-Computer-Interaction applications.

References

[1] Y. Xing and J. Zhu. Deep learning-based action recognition with 3d skeleton: a survey, 2021.

[2] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.



(a) (b)

Fig. 5: Confusion matrices on Briareo of (a) our approach and (b) ST-GCN [2]

[3] W. Zhang, Z. Lin, J. Cheng, C. Ma, X. Deng, and H. Wang. Sta-gcn: two-stream graph convolutional network with spatial–temporal attention for hand gesture recognition. *The Visual Computer*, 36(10):2433–2444, 2020.

[4] Q. De Smedt, H. Wannous, J.-P. Vandeborre, J. Guerry, B. Le Saux, and D. Filliat. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. In *3DOR-10th Eurographics Workshop on 3D Object Retrieval*, pages 1–6, 2017.

[5] F. Manganaro, S. Pini, G. Borghi, R. Vezzani, and R. Cucchiara. Hand gestures for the human-car interaction: The briareo dataset. In *International Conference on Image Analysis and Processing*, pages 560–571. Springer, 2019.

[6] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 409–419, 2018.

[7] M. S. Akremi, R. Slama, and H. Tabia. Spd siamese neural network for skeleton-based hand gesture recognition. In *VISIGRAPP (4: VISAPP)*, pages 394–402, 2022.

[8] F. M. Caputo, P. Prebianca, A. Carcangiu, L. D. Spano, and A. Giachetti. Comparing 3d trajectories for simple mid-air gesture recognition. *Computers & Graphics*, 73:17–25, 2018.

[9] Q. De Smedt, H. Wannous, and J.-P. Vandeborre. Skeleton-based dynamic hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2016.

[10] H. Liu, J. Tu, M. Liu, and R. Ding. Learning explicit shape and motion evolution maps for skeleton-based human action recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1333–1337. IEEE, 2018.

[11] J. Hou, G. Wang, X. Chen, J.-H. Xue, R. Zhu, and H. Yang. Spatial-temporal attention res-tcn for skeleton-based dynamic hand gesture recognition. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

[12] J. Liu, Y. Liu, Y. Wang, V. Prinet, S. Xiang, and C. Pan. Decoupled representation learning for skeleton-based gesture recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5751–5760, 2020.

[13] X. S. Nguyen, L. Brun, O. Lézoray, and S. Bougleux. A neural network based on spd manifold learning for skeleton-based hand gesture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12036–12045, 2019.

[14] Z. Huang and L. Van Gool. A riemannian network for spd matrix learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[15] Z. Huang, J. Wu, and L. Van Gool. Building deep networks on grassmann manifolds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[16] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[18] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3595–3603, 2019.

[19] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Non-local graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:1805.07694*, 1(2):3, 2018.

[20] L. Shi, Y. Zhang, J. Cheng, and H. 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.

[21] R. Li, S. Wang, F. Zhu, and J. Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[22] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks. *IEEE Transactions on Image Processing*, 29:9532–9545, 2020.

[23] Y. Chen, L. Zhao, X. Peng, J. Yuan, and D. N. Metaxas. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. *arXiv preprint arXiv:1907.08871*, 2019.

[24] C. Plizzari, M. Cannici, and M. Matteucci. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding*, 208:103219, 2021.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[27] M. Y. Park and T. Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007.

[28] C. Cortes, M. Mohri, and A. Rostamizadeh. L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*, 2012.

[29] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.