SignGraph: A Sign Sequence is Worth Graphs of Nodes

Shiwei Gan[†] Yafeng Yin[†*] Zhiwei Jiang[†] Hongkai Wen[‡] Lei Xie[†] Sanglu Lu[†]
[†] State Key Laboratory for Novel Software Technology, Nanjing University, China
[‡] Department of Computer Science, The University of Warwick, UK
sw@smail.nju.edu.cn {yafeng,jzw,lxie,sanglu}@nju.edu.cn hongkai.wen@warwick.ac.uk

Abstract

Despite the recent success of sign language research, the widely adopted CNN-based backbones are mainly migrated from other computer vision tasks, in which the contours and texture of objects are crucial for identifying objects. They usually treat sign frames as grids and may fail to capture effective cross-region features. In fact, sign language tasks need to focus on the correlation of different regions in one frame and the interaction of different regions among adjacent frames for identifying a sign sequence. In this paper, we propose to represent a sign sequence as graphs and introduce a simple yet effective graph-based sign language processing architecture named SignGraph, to extract cross-region features at the graph level. SignGraph consists of two basic modules: Local Sign Graph (LSG) module for learning the correlation of ***intra-frame cross-region*** *features in one frame and Temporal Sign Graph (TSG) module for tracking the interaction of* ***inter-frame cross-region*** *features among adjacent frames. With LSG and TSG, we build our model in a multiscale manner to ensure that the representation of nodes can capture cross-region features at different granularities. Extensive experiments on current public sign language datasets demonstrate the superiority of our SignGraph model. Our model achieves very competitive performances with the SOTA model, while not using any extra cues. Code and models are available at: https://github.com/gswycf/SignGraph.*

1. Introduction

Computer vision technology and natural language processing technology have greatly advanced sign language (SL) research, including Sign Language Recognition (SLR) and Sign Language Translation (SLT). The former task aims at recognizing an isolated sign/continuous signs as a corresponding gloss or gloss sequence, while the latter task aims at translating continuous signs into spoken language [3]. In the modern sign language processing models (SLR models and SLT models), one of the key points of these deep learning based methods is to learn *sign-related* features through training with extensive sign video data. As a matter of course, 2D CNNs and 3D CNNs are used to be the de-facto standard visual backbones [7, 10] for extracting local sign frame features, while 1D CNN, LSTM and Transformer [3, 11, 37] are used as the temporal module to capture dynamic changes of sign frames.

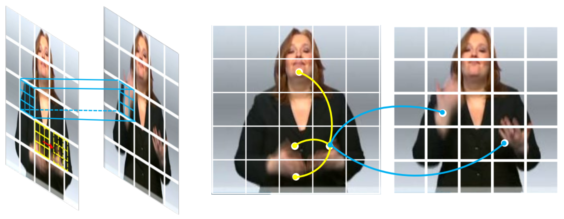
Figure 1. Comparison of feature extraction between convolutional network and our graph convolutional network.

However, as shown in Figure 1, the widely-used CNN-based backbones that are mainly migrated from image recognition and object detection tasks, treat the sign frame/sign clips as grids and adopt 'kernels' to extract local features by sliding across sign frames. These CNNs are good at capturing contour-based representation and texture-based representation [12], but they may fail to capture the explicit collaboration of signs in different regions. In fact, sign language tasks need to focus on manual and non-manual features, especially the collaboration of these cues in different regions [35]. Besides, temporal modules like LSTM and Transformer layer usually regard a sign frame as a whole region and learn the overall variance of whole frames, while ignoring the interaction of subregions in adjacent frames. Other temporal methods like 1D CNN can only focus on the same regions in adjacent frames, which may fail to track dynamic motions across different regions. In fact, the dynamic movements in the body, hands, and face which are reflected in the same/different regions between adjacent frames, are primary elements of identifying a sign for sign language tasks.

In order to make up for the shortcomings of typical CNN backbones explicitly or implicitly and allow the model to learn robust sign-related features, exiting work committed to designing various CNN-based backbones by injecting do-

[*]Yafeng Yin is the corresponding author.

main knowledge *i.e.*, skeleton [11], depth images [30] or local areas [37]. Besides, some work trained their backbones with more training samples by back translation [36], cross modality augmentation [28] and pre-training models [6], while others added extra constraints on backbones by using knowledge distillation [16], contrastive learning [10].

Different from previous work, we build a simple yet effective sign graph neural network (SignGraph for short) for continuous sign language recognition (CSLR) task, in which we take sign frames as graphs of nodes and build dynamic graphs to learn cross-region features. The main intuitions are: (1) Sign-related features involve the correlation of different regions in one frame, such as face region and hand regions. Thus, we propose a local sign graph LSG module by dynamically building graphs based on nodes in each frame, to learn intra-frame cross-region features. (2) The dynamic movements in the body, hands, and face are reflected in different regions among adjacent frames. Thus we propose a temporal sign graph TSG module by dynamically connecting sign-related regions based on nodes in adjacent frames, to learn inter-frame cross-region features. (3) Considering that patches with a single size may not capture local features effectively, we build our SignGraph in a multiscale manner by merging small patches into larger-size patches, and use different-size patches to capture cross-region features at different granularities. We make the following contributions:

- Different from CNN-based SL models, we propose a graph-based CSLR model (SignGraph), which processes sign frames as a set of nodes and dynamically builds graphs from one frame and adjacent frames to capture cross-region features related to signs, thus can aggregate sign-related features from different regions for better feature representation.
- We propose a new local sign graph module LSG that dynamically builds edges based on node features to learn the correlation of cross-region features in one frame, and a new temporal sign graph module TSG that dynamically connects sign-related regions to learn the interaction of cross-region features among adjacent frames.
- We build our model in a multiscale manner with different patch sizes to ensure that the representation of nodes can capture cross-region features at different granularities.
- The extensive experiments on CSLR task demonstrate the superiority of our SignGraph model. Our model achieves very competitive performances with the SOTA model, while not using any extra cues.

2. Related Work

We mainly review CSLR technologies and graph convolutional networks in sign language.

Continuous Sign Language Recognition. The goal of CSLR is to interpret the continuous sign sequence as corresponding gloss sequence [24], thus the core of current CSLR tasks is to identify signs. The widely adopted CNN-based backbones in CSLR which are migrated from the image recognition task, are good at capturing the texture and contours of objects [12]. However, identifying a sign needs to focus on the correlation of different regions (*e.g.*, hand regions and face region) in one frame and the interaction of the same or different regions (*e.g.*, hand motions, facial expression changes) among adjacent frames. As a matter of course, to extract robust sign-related features with CNN-based backbones, some sign language models introduced prior knowledge (or expert knowledge) to guide the model to focus on sign-related features. For example, extra information like local areas [2, 4], skeletons [4, 7, 38] and depth images [30] was manually injected into model. However, in these models, on the one hand, to obtain such extra knowledge, additional algorithms need to be introduced, which will cause additional computational overhead. On the other hand, integrating such knowledge will lead to a complex network structure. Besides, some work turned its attention to obtaining more sign language samples by back translation [36], cross modality augmentation [28], and pre-training with other sign language related datasets [6, 7]. While others were dedicated to designing complex backbones with attention mechanism [18] or adding extra constraints on CNN backbones by adopting knowledge distillation [16, 26], contrastive learning [10].

Graph Convolutional Network. Compared with sequences and grids, graphs are more flexible data structures that model the relationships of a set of objects with nodes and edges. In order to learn representation from the graph, graph convolutional network (GCN) [22] was proposed, and it has been widely applied in action recognition tasks [34] and has also been extended to sign language tasks [11, 17, 21]. To apply GCN to SL tasks, the existing models often relied on extracted skeleton, whose connections between joints make skeleton data naturally fit GCN. Specifically, some work utilized GCN to guide CNN-based backbones to extract skeleton-related features [11, 27]. While others ignored RGB features and applied GCN to explore sign features directly with skeleton [20, 21]. Recently, in addition to applying GCN with skeletons, the usage of applying GCN on images emerged. Specifically, VIG [15] represented an image as a graph structure to capture irregular and complex objects, and demonstrated the effectiveness of graph representation of images. Inspired by VIG, we find that cross-region features are important for identifying a sign and can be captured by graphs. Thus unlike the previous work which adopted CNN-based backbones or relied on skeletons to apply GCN, we directly treat sign frames as graphs of nodes and design a simple GCN-based framework

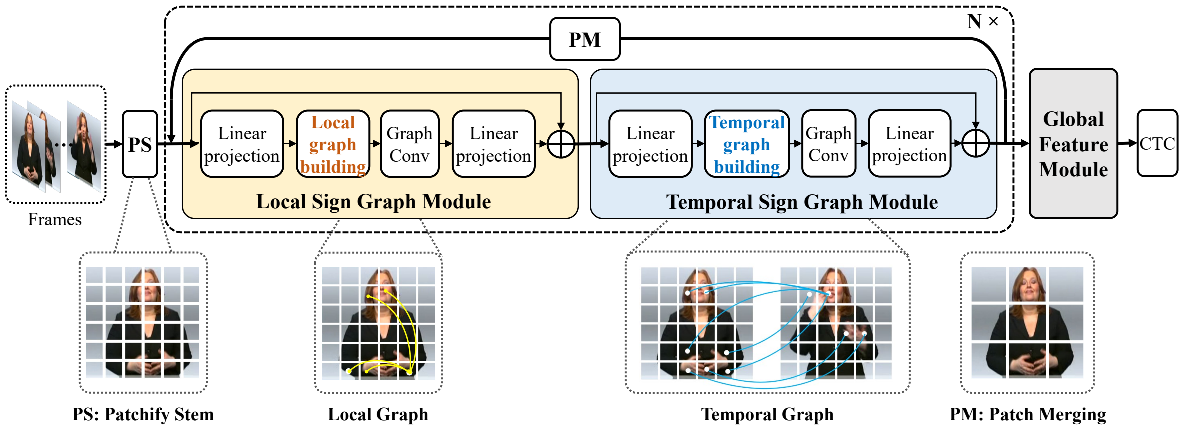

Figure 2. The proposed SignGraph architecture.

for learning sign-related features.

3. Method

In this section, we first give an overview of the proposed graph-based sign language processing architecture. Then we describe the proposed Local Sign Graph LSG module for capturing intra-frame cross-region features and Temporal Sign Graph TSG module for capturing inter-frame cross-region features. Finally, with LSG and TSG, we describe our multiscale SignGraph for CSLR.

3.1. Overall Framework

For a sign language video f=$\{f_i\}_{i=1}^{\theta}$ with θ frames, the target of CSLR model is to get a recognized gloss sequence $g = \{g_i\}_{i=1}^{\vartheta}$ with ϑ glosses based on input f.

As shown in Figure 2, our SignGraph model consists of two key modules, *i.e*. Local Sign Graph LSG module and Temporal Sign Graph TSG module. First, we convert each frame f_i into N patches with a patchify stem. Then each frame f_i is represented by a set of patches (*i.e*., nodes) $v_i = \{v_{ij}\}_{j=1}^{N}$. Second, by learning the correlation between each node and connecting K nearest neighbors for each node, we build a dynamic local graph G_i^l for each frame f_i. Then a local graph convolutional layer is leveraged to capture cross-region features within each frame. Third, by connecting sign-related regions between two adjacent frames, we build a dynamic temporal graph G_i^t for adjacent frames f_i and f_{i+1}. Then, a temporal graph convolutional layer is adopted to track the interaction of cross-region features among adjacent frames.

Further, we merge patches into larger size patches by downsampling frames with a factor of 2, then each frame f_i can be represented by another set of patches (nodes) $\{\bar{v}_{ij}\}_{j=1}^{N/4}$. Following that, we repeatedly adopt LSG module and TSG module to learn cross-region features with different-size patches. After that, following previous sign language models [10, 26], we adopt a global feature module to learn global changes of whole frames. Finally, a classifier and a widely-used CTC loss [13] are adopted to predict the probability $p(g|f)$ of target gloss sequence.

3.2. Sign Graph Learning

Patchify Stem. In order to build graphs based on sign frames instead of skeleton data, we need to convert RGB data into nodes. Following previous work [9, 32], for each frame $f_i \in \mathbb{R}^{[H\times W\times 3]}$ with height H and width W, we first convert it into a number of patches (or nodes) $v_i = \{v_{ij}\}_{j=1}^{N}$ with a patchify stem $\mathcal{PS}$.

$$\mu_i = \{\mu_{ij}\}_{j=1}^{N} = \mathcal{PS}(f_i) \tag{1}$$

Here, v_{ij} represents the jth node in the i-th frame f_i, $\mu_{ij} \in \mathbb{R}^D$ denotes node features of v_{ij}, D is the feature dimension, $N = HW/P^2$ is the number of patches, and P is the patch size.

Local Sign Graph Learning. For a single frame in the sign video, sign language models should focus on the correlation of different regions, such as face region and hand regions. In order to learn intra-frame cross-region features, we need to build a graph for frame f_i, in which regions with correlation (*e.g*., hand regions and face region) should be connected.

As shown in Figure 3, for a set of nodes v_i, we first adopt a linear projection layer with learnable weights W_1^l to map node features μ_i to the space where the distance function is applied: $\mu_i' = \mu_i W_1^l$. Then we adopt K-Nearest Neighbors (KNN) algorithm to find top K_l nearest neighbors $\mathcal{N}_{\mathsf{K}_l}(v_{ij})$ for each node based on distance function $\mathcal{DIS}$ [15].

$$\mathcal{DIS}(v_{ij}, v_{ik}) = \sqrt{\sum(\mu_{ij}' - \mu_{ik}')^2} \tag{2}$$

$$\mathcal{N}_{\mathsf{K}_l}(v_{ij}) = \{v_{ik}|v_{ik} \in TopK(\{\mathcal{DIS}(v_{ij}, v_{ik})\}_{k=1}^{N})\} \tag{3}$$

Here, $TopK$ function outputs the top K nearest neighbors for v_{ij} based on node distances. For each node v_{ij},

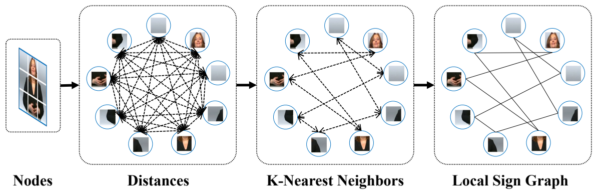

Figure 3. Graph construction in local sign graph module.

we add an edge $e(v_{ij}, v_{ik})$ between v_{ik} and v_{ij}, where $v_{ik} \in \mathcal{N}_{\mathsf{K}_l}(v_{ij})$. Thus, for frame f_i, we can get the edge set $e_i^l = \{e(v_{ij}, v_{ik}), |j \in [1, N], v_{ik} \in \mathcal{N}_{\mathsf{K}_l}(v_{ij})\}$, and build a local graph $G_i^l = \{v_i, e_i^l\}$. After that, we adopt a graph convolutional layer $\mathcal{GCN}_l$ and a linear projection layer with learnable weights W_2^l to aggregate features from cross regions for each frame.

$$\mu_i = \mu_i + \mathcal{GCN}_l(\mu_i', e_i^l)W_2^l \tag{4}$$

The LSG module contains one graph convolutional layer and two linear projection layers. We do not add extra modules (*e.g.*, feed-forward network) since we aim to keep our model simple, and we show that our model can achieve a convincing performance even with the simple design.

Temporal Sign Graph Learning. The dynamic movements in body, hands, and face are essential elements to identify a sign, which are often reflected in different regions among adjacent frames. Hence, we design a temporal graph module, which can dynamically build edges between regions in two adjacent frames and learn inter-frame cross-region features.

For two adjacent frames f_i and f_{i+1}, we have $2N$ nodes $\{v_{ij}\}_{j=1}^{N}$ and $\{v_{(i+1)k}\}_{k=1}^{N}$. Similarly, we first adopt a linear projection layer with weights W_1^t to map node features μ_i to the space where the distance function is applied: $\mu_i''{=}\mu_i W_1^t$. Then we compute distances between v_{ij} and $v_{(i+1)k}$, thus we have a $N{\times}N$ distance matrix M, where $\{M[j,k]{=}\mathcal{DIS}(v_{ij}, v_{(i+1)k})|j \in [1,N], k \in [1,N]\}$. When building the temporal graph, unlike the LSG module where we build edges for each node v_{ij} (*i.e.*, building a dense graph), we only choose top K_t node pairs between two adjacent frames and connect them (*i.e.*, building a sparse graph). To avoid establishing edges between meaningless node pairs, we set a threshold of 0.05 on normalized distance matrix M to filter out patch pairs in the background, since these pairs often have a high and constant similarity (*i.e.*, $M[j,k]{<}0.05$). Then we get the temporal edge set e_i^t between frame f_i and f_{i+1} and temporal graph $G_i^t = \{(v_i, v_{i+1}), e_i^t\}$.

$$e_i^t = \{e(v_{ij}, v_{(i+1)k})|\mathcal{DIS}(v_{ij}, v_{(i+1)k}) \in TopK(M)\} \tag{5}$$

After that, we adopt a temporal graph convolutional layer $\mathcal{GCN}_t$ and another linear projection W_2^t to aggregate cross-region features from adjacent frames, as follows.

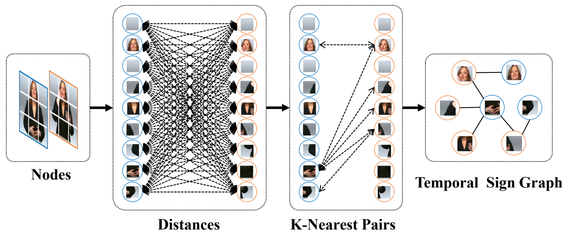

Figure 4. Graph construction in temporal sign graph module.

$$\{\mu_i\}_{i=1}^{\theta} = \{\mu_i\}_{i=1}^{\theta} + \mathcal{GCN}_t\left(\{\mu_i''\}_{i=1}^{\theta}, \{e_i^t\}_{i=1}^{\theta-1}\right) W_2^t \tag{6}$$

The intuition of building the sparse graph for TSG module rather than building the dense graph like LSG module is to avoid the over-smoothing phenomenon [19], which may decrease the distinctiveness of node features.

3.3. Multiscale Sign Graph Convolutional Network

Patches with a fixed window may not capture sign features in one region effectively (*e.g.*, hand regions may be split by patches). Thus, similar to Swin Transformer [25], we propose a multiscale SignGraph, which builds our model in multiscale manners to ensure that the representation of nodes can capture cross-region features at different granularities.

Patch Merging. To produce a multiscale representation, we further design a patch merging module to reduce the number of patches (nodes) and expand the receptive field of patches. Specifically, for N patches (or nodes) with the size of 16 × 16 in frame f_i, the patch merging module applies convolutional layers to downsample frames by a factor of 2. Then we get a set of nodes $\{\bar{v}_{ij}\}_{j=1}^{N/4}$ for frame i, and its corresponding node features $\{\bar{\mu}_{ij}\}_{j=1}^{N/4}$ where $\bar{\mu}_{ij} \in \mathbb{R}^{2D}$.

Multiscale Sign Graph Learning. After getting the set of nodes $\{\bar{v}_{ij}\}_{j=1}^{N/4}$ with a larger size for frame i by patch merging, we adopt the another LSG module with K_l edges for each node in one frame and TSG module with K_t edges for two adjacent frames. For convenience, we name the ith LSG , TSG module as LSG_i, TSG_i, and the corresponding hyperparameters as K_l^i, K_t^i. According to experimental results in Section 4.3, we adopt two LSG, TSG modules in our baseline settings.

4. Experiments

4.1. Datasets

Following the previous work [16, 36], we evaluate our model on three publicly available SL datasets. (1) **PHOENIX14** [23] is a widely used German SL dataset for CSLR with a vocabulary of 1295 glosses from 9 signers. It contains 5672, 540, and 629 weather forecast samples for training, validation, and testing respectively. (2) **PHOENIX14T** [3] is another German SL dataset with both gloss annotations and translation annotations. It contains 7096, 519, and 642 samples from 9 signers for training, validation, and testing respectively, and it has a vocabulary of 1066 different signs for CSLR. (3) **CSL-Daily** is a Chinese SL dataset, which contains 18401, 1077, and 1176 labeled videos from 10 signers for training, validation, and testing respectively, and it has gloss annotations with a vocabulary of 2000 glosses for CSLR.

4.2. Experimental Setting

Here, we describe the baseline settings for our architecture.

Data Preprocessing. For data preprocessing, we follow the existing work [18, 26] and apply data augmentation during training, including: resizing frames to 256×256 pixels, random cropping frames to 224×224 pixels, random horizontal flipping with a probability of 0.5, random temporal scaling (± 20%). During testing, we resize frames to 256×256 pixels and use center cropping to get frames with 224×224 pixels.

Architecture Setting. Our architecture is implemented by PyTorch 1.11. (1) *Patchify stem*: Considering that sign frames have a strong 2D local structure, *i.e.*, spatially neighboring pixels are usually highly correlated, we do not adopt non-overlapping patches, which are obtained by linear projection [9] and lack modeling power for 2D local spatial context [32, 33]. Instead, we obtain initial patches (nodes) embedding by using convolutional-based patchify stem. Specifically, we adopt the first four stages of ResNet18 as our patchify stem. The dimension of initial patches D is set to 512. (2) *Patch merging*: Similar to the patchify stem, we adopt two convolutional blocks with kernel size 3×3 to downsample frames and get different-size patches after LSG and TSG modules. (3) *Sign Graph learning*: There are two LSG and TSG modules in our baseline settings, and the initial K_l^1, K_l^2 in LSG modules are set to 4 and initial K_t^1, K_t^2 in TSG modules are set to 49 (49 is the number of nodes in TSG_2). We adopt the implementation of GCNConv [22] as initial graph convolutional layer (GCN layer). (4) *Distance function*: We adopt Euclidean distance to measure the distance between two nodes in the baseline setting. (5) *Global feature module*: The Global feature module is the combination of two 1D convolution blocks, 2-layer BiLSTM with hidden size 1024 for global feature modeling, and a fully connected layer for final prediction.

Training Setting. For a fair comparison, we adopt the same training settings as in previous work [26]. We adopt the Adam optimizer with a weight decay of 0.0001 to train our model for 40 epochs on 1 GeForce RTX 3090 GPU. The initial learning rate is 0.0001 with a decay factor of 0.5 at epoch 20 and 30, and the batch size is set to 4. CTC loss is applied after both 1D convolution blocks and the fully connected layer as loss functions.

LSG_1	TSG_1	LSG_2	TSG_2	Dev		Test	
				WER	Del/Ins	WER	Del/ins
✗	✗	✗	✗	22.3	8.4/2.5	22.2	8.1/2.7
✓				19.2	5.6/2.2	21.0	4.8/2.3
	✓			19.6	5.6/2.1	21.5	5.1/2.5
		✓		19.3	5.3/2.1	20.8	5.8/2.0
			✓	19.5	6.6/1.7	21.2	5.4/2.4
✓	✓			18.4	5.4/1.6	20.1	5.1/2.1
		✓	✓	19.1	5.6/1.8	20.8	4.8/2.2
✓		✓		18.7	5.1 /2.3	20.6	5.2/1.7
	✓		✓	18.6	4.3 /1.8	20.2	5.5/1.7
✓	✓	✓	✓	**18.1**	5.4/1.4	**20.1**	6.1/1.8

Table 1. Ablation study on Phoenix14T dataset.

Evaluation setting. To evaluate our model, we adopt the Word Error Rate (WER) as CSLR performance metric,

$$WER = \frac{\#S + \#I + \#D}{\#N} \quad (7)$$

where $\#S$, $\#I$, $\#D$ denote the minimum number of substitution, insertion and deletion operations needed to convert a predicted gloss sequence to the ground truth. $\#N$ is the number of glosses in ground truth.

4.3. Ablation study

Following the previous work [3], we perform ablation studies on Phoenix14T dataset to verify the effectiveness of the proposed SignGraph.

Effects of Proposed Sign Graph Module. As shown in Table 1, results demonstrate that both the proposed LSG and TSG modules contribute to higher performance for CSLR. Specifically, our baseline with ResNet18 as backbone achieves 22.3% WER on Phoenix14T dev set. When we replace backbone with the designed local sign graph module LSG_1, LSG_2, WER on dev set decreases by 3.1% and 3.0% respectively. When we replace backbone with designed temporal sign graph module TSG_1, TSG_2, WER decreases by 2.7% and 2.8% respectively, which verifies that both intra-frame cross-region features and inter-frame cross-region features are crucial for learning sign-related features.

Then we combine LSG and TSG modules, WER decreases by 3.9% when applying LSG_1 and TSG_1, 3.2% when applying LSG_2 and TSG_2 respectively, which demonstrates that combination of cross-region features in one frame and those among adjacent frames are beneficial for identifying signs. When adopting two stages of LSG and TSG, performance can achieve further improvement (*i.e.*, WER decreases by 4.2%/2.1% on dev/test set).

LSG_1	TSG_1	Dev		Test	
		WER	Del/Ins	WER	Del/ins
Dense	Sparse	**18.4**	5.4/1.6	**20.1**	5.1/2.1
Dense	Dense	20.2	4.9 /2.9	21.8	5.5/2.6
Sparse	Sparse	19.1	5.0 /1.9	21.7	5.3/2.7
Sparse	Dense	20.9	5.4 /2.4	21.9	6.1/2.5

Table 2. Ablation study on graph types.

It shows that leveraging multiscale representation of nodes can capture cross-region features at different granularities.

Effect of Graph Type. In SignGraph, we adopt dense graphs (*i.e.*, connecting top K neighbors for each node) in the LSG module, while adopting sparse graphs (only connecting top K node pairs) in the TSG module. To verify the effectiveness of our design, we test the effect of different combinations of graph types in LSG and TSG modules. For convenience, we only adopt one LSG module and one TSG module. As shown in Table 2, when we adopt the dense graph in TSG module or adopt the sparse graph in LSG module, there is a noticeable performance drop (*i.e.*, WER on dev set rises by 1.8% or 0.7%). Our module with dense graph in LSG and sparse graph in TSG achieves the best recognition performance, which demonstrates the effectiveness of our model.

Effects of Hyperparameters K. In the proposed multiscale SignGraph, there are four hyperparameters, *i.e.*, K_l^1 K_l^2 in LSG modules and K_t^1, K_t^2 in TSG modules. Here, to set proper hyperparameters K for our model, we test recognition performance under different K_l ranging ranging from 2 to 9 and under different K_t ranging from 7 to 91. Since there are many combinations of these hyperparameters, it is impossible to enumerate all possbile values to find the globally optimal setting, thus we fix the other 3 parameters and adjust one to obtain suitable parameter settings. According to Figure 5, we set K_l^1, K_l^2, K_t^1, K_t^2 to 3, 4, 49, 49 respectively based on WER performance on DEV set.

Effects of Different Backbones. To verify the effectiveness of the proposed SignGraph model, we replace our backbone with other patch-based networks (*i.e.*, CvT [33], Swin transformer [25] and VIG [15]). According to Table 3a, simply adopting the current patch-based SOTA backbones will not gain appealing performance on CSLR task, while our proposed SignGraph backbone achieves satisfactory performance by effectively capturing intra-frame cross-region features and inter-frame cross-region features.

Effects of Distance Function. In the SignGraph, we adopt KNN algorithm to find the nearest neighbors based on the distance of two nodes. Here, we show performances with commonly used distance functions including Cosine distance, Chebyshev distance and Euclidean distance in Table 3b. There are subtle performance gaps between different distance functions and we adopt Euclidean distance in our model for better performances.

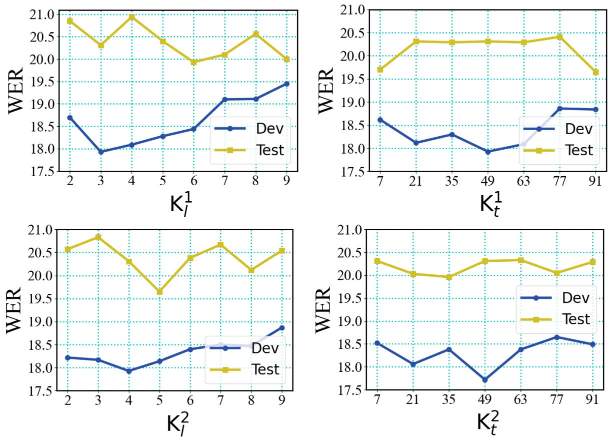

Figure 5. Effects of K_l and K_t.

Effects of GCN Layer. We also test the representative variants of graph convolution, including GATv2Conv [1], SAGEConv [14], GCNConv [22] and EdgeConv [31]. From Table 3c, we can see that our model can achieve excellent performances with different GCN layer, indicating the flexibility of SignGraph model. Among them, EdgeConv achieves the best recognition performance (*i.e.*, 17.8% WER). In the rest of the experiments, we use EdgeConv layer in our module by default unless specifically stated.

Effect of Patch Sizes. Patches with different sizes can capture different local features and further affect model performance. To test the effect of patch sizes, we adopt only one LSG module and one TSG module. As shown in Table 3d, model with a smaller patch size (*i.e.*, 8×8) achieves the worst performance, while model with too large patch size may also degenerate local features.

Effect of Multiscale SignGraph. To leverage the scale-invariant property of images [15], PyVIG adopts pyramid architecture that gradually increases patch size from 4 to 32, by gradually shrinking the spatial size of feature maps. Similarly, we also gradually add LSG and TSG modules in the early stage of the patchify stem and add patch merging module at the end of each TSG module for downsampling. In regard to the added LSG and TSG modules in each stage, we also choose appropriate hyperparamters K_l,K_t for them through extensive experiments. As shown in Table 3e, adding more LSG and TSG modules does not always bring more performance gain, thus we adopt two LSG and TSG modules that increase patch size from 16 to 32 in our model.

Effect of DropEdge. DropEdge [29] tackles over-smoothing and over-fitting problems for dense graph by randomly removing a certain number of edges from the input graph at each training epoch. Considering that we

BackBone	WER	Del	Ins
CvT [32]	39.2	15.3	0.9
SwinT [25]	45.4	16.3	1.3
PyVIG [15]	35.4	12.1	1.3
Ours	**17.9**	5.1	1.5

(a) **Backbone**. Comparison of different backbones.

Distance	WER	Del	Ins
Cosine	18.0	5.0	1.5
Chebyshev	18.2	4.8	1.6
Euclidean	**17.9**	5.1	1.5

(b) **Distance function**. Comparison of distance functions.

GraphConv	WER	Del	Ins
GATv2Conv [1]	18.6	5.2	1.8
SAGEConv [14]	18.0	4.8	1.7
GCNConv [22]	17.9	5.1	1.5
EdgeConv [31]	**17.8**	5.0	1.6

(c) **Graph Convolution** Effect of different GCN layers.

PatchSize	WER	Del	Ins
8	19.1	4.8	2.6
16	**18.4**	5.4	1.6
32	18.7	5.7	1.7

(d) **Patch size**. Comparison of different patch sizes with one LSG and TSG modules.

Stages	WER	Del	Del
16→32	**17.8**	5.0	1.6
8→16→32	18.4	5.3	1.7
4→8→16→32	18.1	4.7	1.9

(e) **Multiscale SignGraph**. Effect of the number of stages in multiscale SignGraph.

DropRate	WER	Del	Ins
0	**17.8**	5.0	1.6
15%	19.2	6.7	1.3
30%	21.3	5.9	1.7

(f) **Drop edge**. Adding DropEdge [29] in the local graph does not improve performance.

Table 3. **Ablation experiments of SignGraph on Phoenix14T dev set.**

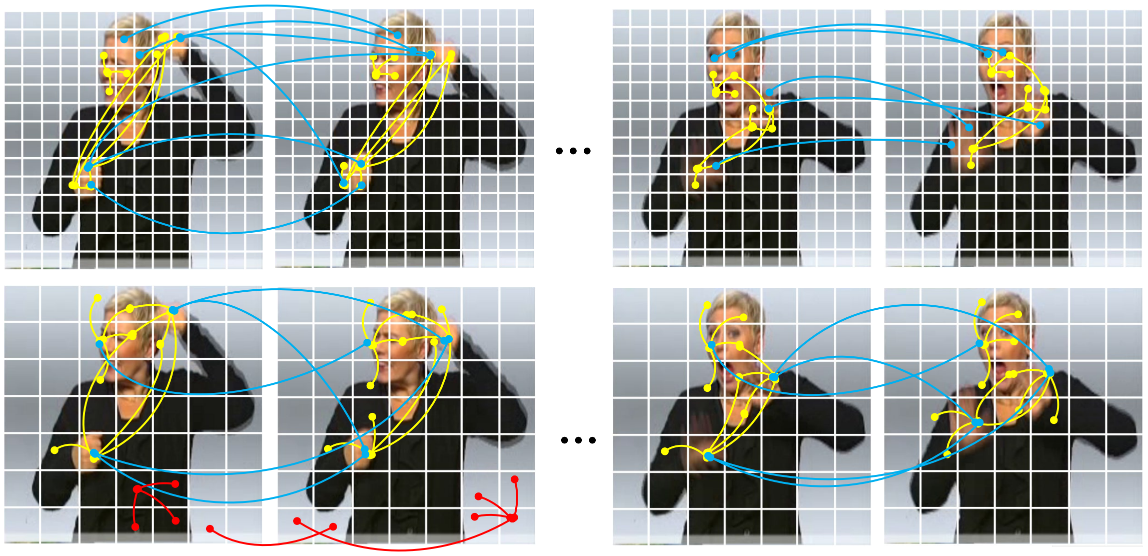

Figure 6. Visualization of graph construction of LSG_1 and TSG_1 in the first row, LSG_2 and TSG_2 in the second row.The graph in LSG module is shown in yellow, and the graph in TSG is shown in blue. We also show some 'unimportant' edges between nodes in the background with red color.

build dense graphs for LSG module, we also add DropEdge with different drop rates to LSG module and show the recognition performance in Table 3f. However, adding DropEdge does not improve our model performance, which means that our model can achieve excellent performance without relying on external modules, and can handle over-smoothing and over-fitting problems well.

Visualization of SignGraph. To verify whether our model can effectively capture sign-related features, we select a sign video from Phoenix14T test set and visualize the constructed graph structure of LSG and TSG modules in both two stages. In Figure 6, we show the graphs of two stages and we only show part of the edges for a better display. In the LSG module, our model can link the nodes with similar content and semantic representation (*e.g.*, hand regions and face regions) for extracting better intra-frame cross-region features. In the TSG module, edges can be built for tracking dynamic changes in gestures and facial expressions, thus can better identify a sign with inter-frame cross-region features. We also show some 'unimportant' edges between nodes in the background with red color. As can be seen, background nodes in LSG are naturally connected to their neighboring nodes, and there are still a few background nodes connected in TSG. Fortunately, nodes in the background will not 'disturb' nodes in sign-related regions, which demonstrates the effectiveness of our model.

4.4. Comparisons

Evaluation on Phoenix14T dataset. As shown in Table 4, we compare our model with existing models on CSLR performance, and we provide both the performances on the validation set (*i.e.*, 'DEV') and test set (*i.e.*, 'TEST'). Most of the current models adopt existing CNN-based backbones, and achieve convincing performances by injecting extra cues [7, 35, 38], adding extra constraints [10, 16, 26] or introducing attention mechanism [18]. The GCN-based

Model	Backbone	Extra cues				Phoenix				PhoenixT	
						DEV		TEST		DEV	TEST
		F/M	H	S	P	WER	del/ins	WER	del/ins	WER	WER
STMC [35] (AAAI'20)	VGG11	✓	✓	✓		21.1	7.7/3.4	20.7	7.4/2.6	19.6	21.0
C2SLR [38](CVPR'22)	ResNet18			✓		20.5	-/-	20.4	-/-	20.2	20.4
TwoStream [7](NeurIPS'22)	S3D	✓	✓	✓	✓*	18.4	-/-	18.8	-/-	17.7	19.3
SLT [5] (CVPR'20)	GooleNet				✓	-	-	-	-	24.6	24.5
TwoStream [7] (NeurIPS'22)	S3D				✓*	22.4	-/-	23.3	-/-	21.1	22.4
VAC [26] (ICCV'21)	ResNet18				✓	21.2	7.9/2.5	22.3	8.4/2.6	-	-
SMKD [16] (ICCV'21)	ResNet18				✓	20.8	6.8/2.5	21.0	6.3/2.3	20.8	22.4
CorrNet [18] (CVPR'23)	ResNet18				✓	18.8	5.6/2.8	19.4	5.7/2.3	18.9	20.5
FCN [8] (ECCV'20)	customed					23.7	-/-	23.9	-/-		
Contrastive [10] (IJCAI'23)	ResNet18					19.6	5.1/2.7	19.8	5.8/3.0	20.0	20.1
HST-GNN [21] (WACV'22)	Customized(CNN+GCN)	✓	✓	✓	✓	19.5	-/-	19.8	**-/-**	19.5	19.8
CoSign [20] (ICCV'23)	ST-GCN(GCN)	✓	✓	✓		19.7	-/-	20.1	**-/-**	19.5	20.1
SignGraph	Customized(GCN)				✓	18.4	5.6/1.8	20.1	5.4/2.1	18.3	20.0
MultiSignGraph	Customized(GCN)				✓	18.2	4.9/2.0	19.1	5.3/1.9	17.8	19.1

Table 4. Comparison of CSLR performance on Phoenix14 and Phoenix14T datasets. (F: face, M: mouth, H: hands, S: skeleton, P: pretraining backbone with ImageNet, ✓*: pretraining on other SL-related datasets.)

Model	Backbone	Extra cues		DEV		TEST	
		S	P	WER	del/ins	WER	del/ins
SLT [5](CVPR'20)	GoogleNet		✓	33.1	10.3/4.4	32.0	9.6/4.1
TwoStream [7](NeurIPS'22)	S3D	✓	✓*	25.4	-/-	25.3	-/-
TwoStream [7](NeurIPS'22)	S3D		✓*	28.9	-/-	28.5	-/-
BN-TIN [36](CVPR21)	GoogLeNet		✓	33.6	13.9/3.4	33.1	13.5/3.0
CorrNet [18] (CVPR'23)	ResNet18		✓	30.6	-/-	30.1	-/-
CoSign [20] (ICCV'23)	ST-GCN(GCN)	✓		28.1	-/-	27.2	**-/-**
SignGraph	customized(GCN)		✓	28.4	8.8/2.4	27.4	8.2/2.1
MultiSignGraph	customized(GCN)		✓	27.3	7.9/2.3	26.4	7.8/2.1

Table 5. Comparison of CSLR performance on CSL-daily dataset.

model CoSign [20] mainly relies on pre-processed fine-grained skeleton data, and achieves 19.5%, 20.1% WER on dev, test set. While HST-GCN [21] adopts both CNN-based backbone and GCN-based backbone to extract RGB features and skeleton features respectively, achieving 19.5%, 19.8% WER on dev, test set. It is worth mentioning that the state-of-the-art (SOTA) model TwoStream utilizes both RGB features and fine-grained skeleton features (*i.e.*, keypoints in hands, body and face), and pretrains its backbone on other SL-related datasets, achieving 17.7%, 19.3% WER on dev, test set respectively. Without using any extract cues, our best model still achieves comparable performance with TwoStream on dev set. While on test set, our model even outperforms TwoStream model by 0.2% WER.

Evaluation on Phoenix14 dataset. We also compare our model with existing models of CSLR performance on Phoenix14 dataset. As shown in Table 4, with the simple yet powerful multiscale SignGraph, our model can achieve the best performance (*i.e.*, 18.2% WER) on dev set and a comparable performance with SOTA model on test set, which demonstrates the effectiveness of our model.

Evaluation on CSL-daily dataset. We also provide CSLR performance comparisons with other methods on CSL-daily dataset. As shown in Table 5, the WER of our model on dev and test set is 27.3% and 26.4%, respectively. Our model outperforms most existing models, and achieves a comparable performance with TwoStream model.

5. Conclusion

In this paper, we propose a simple yet effective SignGraph architecture, which represents a sign sequence as a graph structure to extract sign-related features at the graph level. Specifically, to learn the correlation of cross-region features in one frame, we propose a Local Sign Graph LSG module, which dynamically builds edges between nodes in one frame and aggregates intra-frame cross-region features. To track the interaction of cross-region features among adjacent frames, we propose a Temporal Sign Graph TSG module, which dynamically builds edges between nodes among adjacent frames and captures inter-frame cross-region features. Besides, we also build our model in multiscale manners with different patch sizes to ensure that the representation of nodes can capture cross-region features at different granularities. The extensive experiments on public datasets demonstrate the superiority of the proposed SignGraph, which achieves comparable performances with the SOTA model without using any extra cues.

6. Acknowledgments

This work is supported by National Key Research and Development Program of China under Grant No. 2022YFB3303900; Jiangsu Provincial Key Research and Development Program under Grant BE2020001-4; National Natural Science Foundation of China under Grant Nos. 62172208, 62272216. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021. 6, 7

[2] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, and Richard Bowden. Subunets: End-to-end hand shape and continuous sign language recognition. In *ICCV*, pages 3075–3084. IEEE, 2017. 2

[3] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, Hermann Ney, and Richard Bowden. Neural sign language translation. In *CVPR*, pages 7784–7793, 2018. 1, 5

[4] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Multi-channel transformers for multi-articulatory sign language translation. In *ECCV*, pages 301–319. Springer, 2020. 2

[5] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. Sign language transformers: Joint end-to-end sign language recognition and translation. In *CVPR*, pages 10023–10033, 2020. 8

[6] Yutong Chen, Fangyun Wei, Xiao Sun, Zhirong Wu, and Stephen Lin. A simple multi-modality transfer learning baseline for sign language translation. In *CVPR*, pages 5120–5130, 2022. 2

[7] Yutong Chen, Ronglai Zuo, Fangyun Wei, Yu Wu, Shujie Liu, and Brian Mak. Two-stream network for sign language recognition and translation. *Advances in Neural Information Processing Systems*, 35:17043–17056, 2022. 1, 2, 7, 8

[8] Ka Leong Cheng, Zhaoyang Yang, Qifeng Chen, and Yu-Wing Tai. Fully convolutional networks for continuous sign language recognition. In *ECCV*. Springer, 2020. 8

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 3, 5

[10] Shiwei Gan, Yafeng Yin, Zhiwei Jiang, Kang Xia, Lei Xie, and Sanglu Lu. Contrastive learning for sign language recognition and translation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 763–772. 1, 2, 3, 7, 8

[11] Shiwei Gan, Yafeng Yin, Zhiwei Jiang, Lei Xie, and Sanglu Lu. Skeleton-aware neural sign language translation. In *MM*, pages 4353–4361, 2021. 1, 2

[12] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. 1, 2

[13] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376, 2006. 3

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 6, 7

[15] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. Vision gnn: An image is worth graph of nodes. *Advances in Neural Information Processing Systems*, 35:8291–8303, 2022. 2, 3, 6, 7

[16] Aiming Hao, Yuecong Min, and Xilin Chen. Self-mutual distillation learning for continuous sign language recognition. In *ICCV*, pages 11303–11312, 2021. 2, 4, 7, 8

[17] Hezhen Hu, Weichao Zhao, Wengang Zhou, and Houqiang Li. Signbert+: Hand-model-aware self-supervised pre-training for sign language understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2

[18] Lianyu Hu, Liqing Gao, Zekang Liu, and Wei Feng. Continuous sign language recognition with correlation network. In *CVPR*, pages 2529–2539, 2023. 2, 5, 7, 8

[19] Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Tackling over-smoothing for general graph convolutional networks. *arXiv preprint arXiv:2008.09864*, 2020. 4

[20] Peiqi Jiao, Yuecong Min, Yanan Li, Xiaotao Wang, Lei Lei, and Xilin Chen. Cosign: Exploring co-occurrence signals in skeleton-based continuous sign language recognition. In *ICCV*, pages 20676–20686, 2023. 2, 8

[21] Jichao Kan, Kun Hu, Markus Hagenbuchner, Ah Chung Tsoi, Mohammed Bennamoun, and Zhiyong Wang. Sign language translation with hierarchical spatio-temporal graph neural network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3367–3376, 2022. 2, 8

[22] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2, 5, 6, 7

[23] Oscar Koller, Jens Forster, and Hermann Ney. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *CVIU*, 141:108–125, 2015. 4

[24] Oscar Koller, Sepehr Zargaran, and Hermann Ney. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent cnn-hmms. In *CVPR*, pages 4297–4305, 2017. 2

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 4, 6, 7

[26] Yuecong Min, Aiming Hao, Xiujuan Chai, and Xilin Chen. Visual alignment constraint for continuous sign language recognition. In *ICCV*, pages 11542–11551, 2021. 2, 3, 5, 7, 8

[27] Maria Parelli, Katerina Papadimitriou, Gerasimos Potamianos, Georgios Pavlakos, and Petros Maragos. Spatio-temporal graph convolutional networks for continuous sign language recognition. In *ICASSP*. IEEE, 2022. 2

[28] Junfu Pu, Wengang Zhou, Hezhen Hu, and Houqiang Li. Boosting continuous sign language recognition via cross modality augmentation. In *MM*, pages 1497–1505, 2020. 2

[29] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. 6, 7

[30] Shengeng Tang, Dan Guo, Richang Hong, and Meng Wang. Graph-based multimodal sequential embedding for sign language translation. *TMM*, 2021. 2

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019. 6, 7

[32] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22–31, 2021. 3, 5, 7

[33] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *Advances in neural information processing systems*, 34:30392–30400, 2021. 5, 6

[34] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 2

[35] Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. Spatial-temporal multi-cue network for continuous sign language recognition. In *AAAI*, 2020. 1, 7, 8

[36] Hao Zhou, Wengang Zhou, Weizhen Qi, Junfu Pu, and Houqiang Li. Improving sign language translation with monolingual data by sign back-translation. In *CVPR*, pages 1316–1325, 2021. 2, 4, 8

[37] Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. Spatial-temporal multi-cue network for sign language recognition and translation. *TMM*, 2021. 1, 2

[38] Ronglai Zuo and Brian Mak. C2slr: Consistency-enhanced continuous sign language recognition. In *CVPR*, pages 5131–5140, 2022. 2, 7, 8