



Towards Real-Time Sign Language Recognition and Translation on Edge Devices

Shiwei Gan
State Key Laboratory for Novel
Software Technology, Nanjing
University
China
sw@smail.nju.edu.cn

Yafeng Yin*
State Key Laboratory for Novel
Software Technology, Nanjing
University
China
yafeng@nju.edu.cn

Zhiwei Jiang
State Key Laboratory for Novel
Software Technology, Nanjing
University
China
jzw@nju.edu.cn

Lei Xie
State Key Laboratory for Novel
Software Technology, Nanjing
University
China
lxie@nju.edu.cn

Sanglu Lu
State Key Laboratory for Novel
Software Technology, Nanjing
University
China
sanglu@nju.edu.cn

ABSTRACT

To provide instant communication for hearing-impaired people, it is essential to achieve real-time sign language processing anytime anywhere. Therefore, in this paper, we propose a Region-aware Temporal Graph based neural Network (RTG-Net), aiming to achieve real-time Sign Language Recognition (SLR) and Translation (SLT) on edge devices. To reduce the computation overhead, we first construct a shallow graph convolution network to reduce model size by decreasing model depth. Besides, we apply structural re-parameterization to fuse the convolutional layer, batch normalization layer and all branches to simplify model complexity by reducing model width. To achieve the high performance in sign language processing as well, we extract key regions based on keypoints in skeleton from each frame, and design a region-aware temporal graph to combine key regions and full frame for feature representation. In RTG-Net, we design a multi-stage training strategy to optimize keypoint selection, SLR and SLT step by step. Experimental results demonstrate that RTG-Net achieves comparable performance with existing methods in SLR or SLT, while greatly reducing the computation overhead and achieving real-time sign language processing on edge devices. Our code is available at <https://github.com/SignLanguageCode/realtimeSLRT>.

CCS CONCEPTS

• **Computing methodologies** → **Activity recognition and understanding**.

*Yafeng Yin is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MM '23, October 29–November 3, 2023, Ottawa, ON, Canada.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0108-5/23/10...\$15.00
<https://doi.org/10.1145/3581783.3611820>

KEYWORDS

Sign Language Translation, Sign Language Recognition, Real-Time, Edge Device

ACM Reference Format:

Shiwei Gan, Yafeng Yin, Zhiwei Jiang, Lei Xie, and Sanglu Lu. 2023. Towards Real-Time Sign Language Recognition and Translation on Edge Devices. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23)*, October 29–November 3, 2023, Ottawa, ON, Canada. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3581783.3611820>

1 INTRODUCTION

Sign language is the primary communication way for deaf people. To bridge the communication gap between deaf people and hearing people, a lot of research work on sign language understanding emerged, including sign language recognition (SLR) [26] and sign language translation (SLT) [4]. Specifically, SLR aims to recognize a series of signs as corresponding gloss sequence. While SLT further advances the understanding of sign language, it aims to translate the sign language into spoken language, i.e., the translated sentence satisfies the grammatical rules and linguistic characteristics of spoken language. Therefore, SLT can be treated as a further step of SLR. When applying an appropriate translation model [52] on recognized gloss sequence of SLR, it is possible to get the translated sentence of SLT.

Whatever for SLR or SLT, the existing work mainly focused on extracting efficient features from videos by 2D convolution [3] or 3D convolution [22]. Compared with 2D convolution, the 3D convolution further takes temporal features into consideration, thus the computation overhead of 3D convolution is often very heavy. However, as shown in Figure 1 (a), the existing work usually paid little attention to computation overhead, and they often worked on powerful servers configured with high-performance GPUs, while difficult to work on devices with ordinary computing power, e.g.,

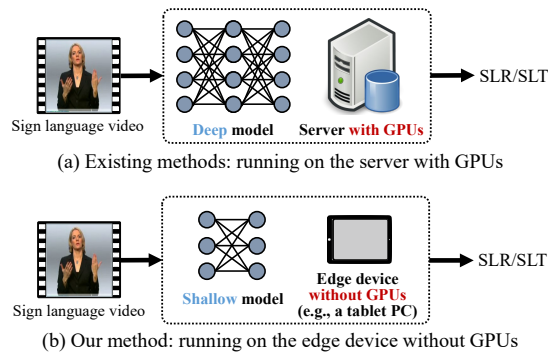


Figure 1: Difference between existing methods and our method on SLR and SLT.

edge devices¹ without GPUs. In fact, to provide instant communications for deaf people anytime anywhere, it is important to achieve real-time sign language processing on portable edge devices (e.g., Microsoft Surface Pro 6), as shown in Figure 1 (b). In regard to the extracted features, they are often consisted of full-frame features [12] and local-area features [56]. However, the prior work tended to extract full-frame and local-area features individually, and concatenated them for final feature representation. The region structure between full frames and local areas, and the temporal interactions among the same regions (e.g., the hand area) in adjacent frames had not been fully studied. Therefore, a real-time sign language processing approach with more efficient representation of region-related features is expected.

To address the above issues, in this paper, we propose a Region-aware Temporal Graph based Network (RTG-Net) for real-time SLR and SLT on edge devices. *First, to achieve real-time sign language processing on edge devices without GPUs*, we design a shallow graph convolution network, which utilizes the lightweight 2D convolutions and graph convolutions to reduce model size by decreasing model depth (i.e., the number of layers in a model). Furthermore, we apply the structural re-parameterization to fuse the convolutional layer, batch normalization layer and all branches, to further simplify model complexity by reducing model width (i.e., number of branches in a model). *Second, to extract efficient features with the shallow network and achieve the good performance for SLR and SLT*, we first extract key regions based on keypoints in skeleton from each frame, to highlight both manual features like hand motion and non-manual features like facial expression in sign language. Then, we combine key regions and full frame (i.e., whole region) by designing a Region-aware Temporal Graph, which preserves the region structure in one frame as well as the region’s correlations among adjacent frames (i.e., along time), to make full use of both spatial and temporal information of signs. Finally, we design a multi-stage optimization strategy for model training to further improve SLR and SLT performance of RTG-Net. We make the following contributions in this paper.

- To the best of our knowledge, this is the first work focusing on real-time video-based SLR and SLT on portable edge

¹Edge devices can encompass various types of devices, e.g., smartphones, tablets, personal computers. In this paper, the edge device refers to Microsoft Surface Pro 6, which can be used as a tablet.

devices, to assist for instant communication between deaf people and hearing people.

- To achieve the real-time requirement, we design a shallow graph convolution network to reduce model size by decreasing model depth. Besides, we adopt the structural re-parameterization to fuse the convolutional layers, batch normalization layers and all branches, to simplify model complexity by reducing model width.
- To extract efficient features for SLR and SLT with lightweight network, we combine the key-region features and full-frame feature by designing a Region-aware Temporal Graph, which preserves the region structure in one frame as well as the region’s correlations among adjacent frames (i.e., along time).
- Extensive experiments on three public datasets demonstrate that our network can achieve real-time SLR and SLT on edge devices (i.e., Microsoft Surface Pro 6) while having comparable performance with existing methods.

2 RELATED WORK

Sign language recognition: Sign language recognition (SLR) can be further classified into two categories, i.e., isolated SLR (ISLR) and continuous SLR (CSLR). The ISLR aims to recognize a sign as a word, it can be treated as sign classification. To achieve this goal, the hand-crafted features [47] and Hidden Markov Model [16, 18] were adopted in early work. Recently, to further improve the feature representation, neural networks like CNN [20, 21] and LSTM [1, 34] were introduced for automatic feature extraction. When moving to CSLR [11, 19, 25, 27, 31, 35, 36, 40, 41, 54], it aims to recognize a series of signs as gloss sequence. To achieve this goal, the traditional methods like DTW-HMM [53] and CNN-HMM [27–29] introduced Hidden Markov Model (HMM) to model sign sequence features. Recently, the deep learning based methods [3, 22, 56] solved CSLR by sequence to sequence learning, which learned the correspondence between sign sequence and gloss sequence. In the work, the Connectionist Temporal Classification (CTC) loss [15] which requires that the source sequence and the target one have the same order was often adopted. However, due to the grammatical differences, the signs in sign language and words in spoken language can be different. Thus the recognized gloss sequence of CSLR may not satisfy the requirement of spoken language.

Sign language translation: Sign language translation (SLT) [8, 9] aims to translate sign language into spoken language. The traditional methods [2] decomposed SLT into two stages, including CSLR and gloss-to-text translation. These two-stage methods need both gloss annotations and sentence annotations, and can be optimized in two stages. In recent years, the deep learning based methods were introduced for SLT in an end-to-end manner. These approaches often adopted encoder-decoder framework [14] and they utilized the encoder to extract features from videos, while using the decoder to generate the spoken language based on extracted features. Specifically, Camgoz et al. [4] first introduced the encoder-decoder network and attention mechanism for end-to-end SLT. Further, Camgoz et al. [6] introduced the transformer networks for joint SLR and SLT. Usually, these approaches utilized the encoder to extract features from videos, while using the decoder to generate the spoken language based on extracted features.

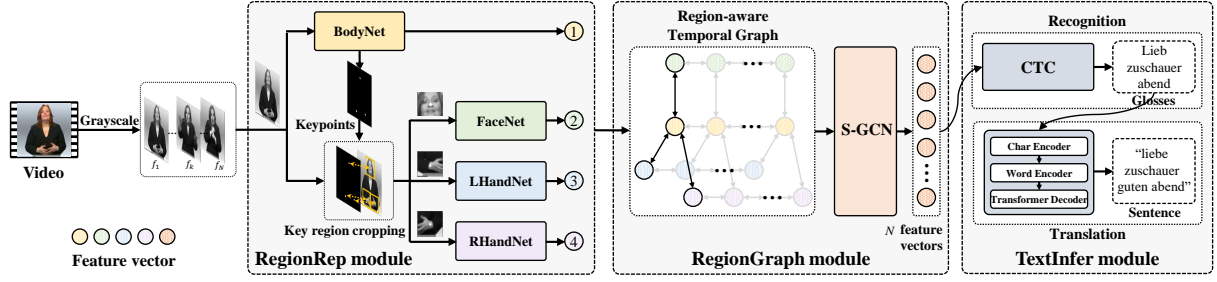


Figure 2: The framework of RTG-Net.

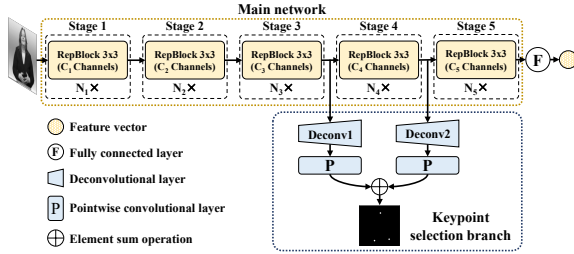


Figure 3: The design of BodyNet. Note: FaceNet, LHandNet and RHandNet have the similar architecture with the main network of BodyNet, while not having the keypoint selection branch.

Whatever for SLR or SLT, to achieve a good performance, these deep learning based approaches usually focused on extracting efficient features, e.g., using 2D convolution to extract spatial features, using 3D convolution to extract spatial-temporal features, combining full-frame and local-area features. However, this work mainly focused on recognition or translation performance, while paying little attention to the computation overhead in SLR or SLT. In fact, due to the heavy computation, the deep learning based method often worked on powerful servers configured with high-performance GPUs, while difficult to work on devices with ordinary computing power, e.g., edge devices without GPUs. Different from the existing work, this paper aims to achieve real-time sign language recognition and translation on edge devices by designing a lightweight network with a good capacity of feature extraction, aiming to provide instant communication for deaf people and hearing people.

3 PROPOSED APPROACH

Given a sign language video $X = \{f_t\}_{t=1}^u$ with u frames, the objective of SLR is to predict the corresponding gloss sequence $G = \{g_t\}_{t=1}^v$ with v glosses, while the goal of SLT is to generate a spoken language sentence $Y = \{w_t\}_{t=1}^\tau$ with τ words based on the glosses G .

As shown in Figure 2, we first convert RGB frames of the input sign video to grayscale frames. Then we design RegionRep module to extract key regions (i.e., face region, left/right hand region) from each frame based on keypoints of skeleton, and design four subnetworks to get feature representation of full frame and key regions. After that, we design the RegionGraph module by constructing a region-aware temporal graph and a graph convolutional network to get the final feature representation for each sign language video.

Table 1: The detailed settings of each subnetwork. (N_i, C_i) means that stage i adopts N_i RepBlocks and outputs C_i channels. Note: LHandNet and RHandNet share the same architecture.

Name	(N_1, C_1)	(N_2, C_2)	(N_3, C_3)	(N_4, C_4)	(N_5, C_5)
BodyNet	(1,64)	(4,64)	(6,128)	(8,256)	(2,512)
FaceNet	(1,32)	(2,32)	(2,64)	(2,128)	(1,512)
L/RHandNet	(1,32)	(2,32)	(4,64)	(8,128)	(1,512)

Finally, we design the TextInfer module, which adopts joint loss for SLR and introduces a tiny translation subnetwork for SLT. It is worth noting that, to reduce parameters and computation overhead, we first reduce model depth by utilizing lightweight 2D convolutions and graph convolutions, then reduce model width by applying the structural re-parameterization to fuse all branches.

3.1 Key Region Extraction and Representation

In sign language, the signs are composed of both manual features like hand motion and non-manual features like facial expression [51]. To emphasize manual features and non-manual features, we additionally extract the key regions (i.e., left/right hand region, face region) from images to enhance feature representation of signs. Specifically, we design the RegionRep module, which first crops key regions from each frame based on human skeleton information, and then extracts features from both full frame and key regions based on four designed subnetworks (i.e., BodyNet, FaceNet, LHandNet, RHandNet).

Key region extraction: To extract key regions (i.e., face region, left/right hand region) from each frame, we first introduce the keypoints (i.e., nose, left wrist and right wrist) in skeleton to locate the center of each key region, and then crop a rectangular region around each center to get key regions.

To get the three keypoints, we design a self-contained branch in BodyNet to get the heatmaps containing keypoints, as shown in Figure 3. When inputting an image (i.e., full frame) to BodyNet, the main network is used to extract features from the full frame, while the keypoint selection branch is used to extract the keypoints from full frame. In regard to the keypoint selection branches, two parallel deconvolutional layers after stage 3 and stage 4 are used to upsample the high-to-low representations [43]. Then, the pointwise convolutional layers and element-sum operations are adopted to generate three heatmaps, where each heatmap contains one keypoint (x_i, y_i) , $i \in [1, 3]$. Specially, a heatmap is a 2D matrix, where the coordinate point with the highest heatvalue is

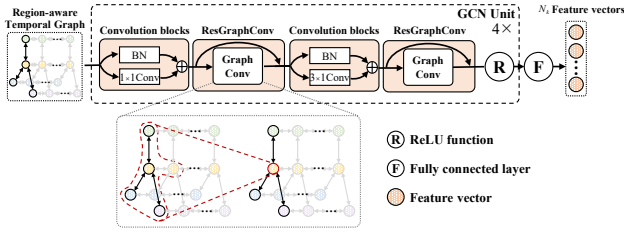


Figure 4: Design of region-aware temporal graph and S-GCN.

the keypoint. After that, we respectively crop a rectangular region $\{(x, y) | (x_i - \frac{w_r}{2}) \leq x \leq (x_i + \frac{w_r}{2}), (y_i - \frac{h_r}{2}) \leq y \leq (y_i + \frac{h_r}{2})\}$ around a keypoint to get three key regions, where w_r, h_r are width and height of cropped region.

Feature representation: To get the feature representation of full frame, face region, hand regions, we accordingly design four subnetworks, i.e., BodyNet, FaceNet, LHandNet, RHandNet. The main framework of each subnetwork is the same, as the main network of BodyNet shown in the top row of Figure 3. The only differences among the four subnetworks are the number of RepBlocks and output channels, as listed in Table 1. In regard to the design of each subnetwork, as shown in Figure 3, it is based on RepVGG [13], which utilizes RepBlock (shown in Figure 5) to extract feature maps from the input image. Then, the extracted feature maps are concatenated, flattened and sent to a fully connected layer to get a feature vector with 256 elements for each key region or full frame.

3.2 Region-Aware Temporal Graph Based Feature Extraction

In addition to the full frame, the key regions are also cropped for feature extraction. For key regions in the same frame, they can combine different regions (i.e., spatial information) to describe a gesture. For the same type of key regions (e.g., left hand regions) in consecutive frames, they can describe the dynamic changes (i.e., temporal information) of an action. To make full use of both spatial and temporal information of signs, we design a novel RegionGraph module, which first constructs a Region-aware Temporal Graph (RTG) by preserving the region structure in one frame as well as the region correlations among consecutive frames, then proposes a shallow graph convolutional network to extract and fuse features from different regions based on RTG.

Region-aware Temporal Graph: To construct the Region-aware Temporal Graph $G=(V, E)$, we treat the full frames, face regions, left/right hand regions as nodes V , while the relationship between nodes are the edges E . In the i th frame, there are four nodes $\{v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4}\}$ corresponding to full frame, face, left/right hand region. For N frames, we get the node set $V = \{v_{ij}, i \in [1, N], j \in [1, 4]\}$. In regard to the edge set, it includes the intra-frame edge set E_a and inter-frame edge set E_e , as shown in Figure 4. For intra-frame edges, the full frame v_{i_1} which has overlap with each key region is selected as the center node and connects with each key region, as shown in Figure 2. Thus we can get the intra-frame edge set $E_a = \{v_{i_p}, v_{i_q} | i \in [1, N], (p, q) \in S\}$ where $S = \{(1, 2), (1, 3), (1, 4), (2, 1), (3, 1), (4, 1)\}$. For the inter-frame edge set, it is consisted of the edges between corresponding nodes in two adjacent frames, thus $E_e = \{v_{i_p}, v_{j_p} | i, j \in [1, N], |i - j| = 1, p \in [1, 4]\}$.

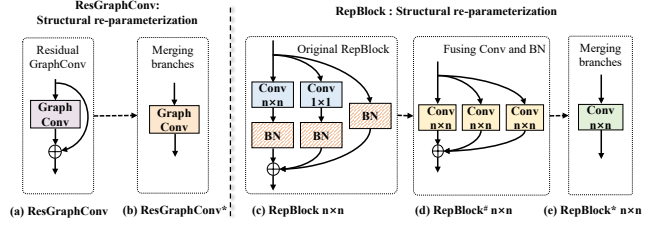


Figure 5: Structural re-parameterization for ResGraphConv and RepBlock.

Feature extraction with Shallow GCN: With the Region-aware Temporal Graph, we design a Shallow Graph Convolution Network (S-GCN) to aggregate the features from different regions. As shown in Figure 4, our S-GCN consists of four GCN units, where each GCN unit consists of two convolution blocks, two residual graph convolution blocks (ResGraphConv) and a ReLU function. Specially, 1×1 convolution block is used for exploiting intra-node features and changing feature dimensions, 3×1 convolution block is adopted for leveraging dynamic changes of the same key regions along time, and ResGraphConv block consisted of a graph convolution layer and a skip connect is designed for node feature propagation and aggregation. After four GCN units, the flatten operation and a fully connected layer are adopted to get the final feature vector $o_i, i \in [1, N]$ for each frame. Specially, the input of S-GCN is the Region-aware Temporal Graph $G=(V, E)$, where the initial representation of each node v_{ij} is the output feature vector from RegionRep module, while the representation of edges is an adjacency matrix $M^e \in \mathbb{R}^{4N \times 4N}$ describing the connections between nodes, as shown in Equation 1. Here, I^v means the input feature vector set of node set, while O means the final feature set and will be sent into the following modules for SLR or SLT.

$$O = S\text{-GCN}(I^v, M^e) \quad (1)$$

3.3 Structural Re-parameterization

Considering the limited computing resource on edge devices, we re-design the structural re-parameterization to simplify all branches in our model, including the firstly-proposed new structural re-parameterization method for residual graph convolution block, as shown in Figure 5. Usually, structure re-parameterization [13] is used to decouple the training stage and inference stage in neural network. In the training stage, we keep the original structures to make the model converge better [46]. While in the inference stage, we fuse all branches to simplify the computation of all modules, where the simplified modules are mathematically equivalent to original modules.

Fusing residual graph convolution block: As shown in Figure 5(a) and 5(b), our ResGraphConv block consists of a graph convolution layer [24] with a skip connection, and it can be formulated as follows,

$$X' = X + \mathcal{GCN}(X) = X + (\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} X \Theta) \quad (2)$$

where X denotes features of node set V , \hat{A} denotes the adjacency matrix of RTG with inserted self-loops, \hat{D} is diagonal degree matrix and Θ is graph convolution weight. To demonstrate the fusion of ResGraphConv block, we transform Equation 2 to Equation 3 from

the aspect of node.

$$\mathbf{x}'_i = \mathbf{x}_i + \mathcal{GCN}(\mathbf{x}_i) = \mathbf{x}_i + (\Theta_1 \mathbf{x}_i + \Theta_2 \sum_{j \in N(i)} e_{j,i} \mathbf{x}_j) \quad (3)$$

where \mathbf{x}_i is feature of node i , $e_{j,i}$ is the edge weight from source node j to target node i , Θ_1, Θ_2 are graph convolution weights, and $N(i)$ is neighbor nodes of node i . Furthermore,

$$\mathbf{x}'_i = (\Theta_1 + \mathbf{I})\mathbf{x}_i + \Theta_2 \sum_{j \in N(i)} e_{j,i} \mathbf{x}_j = \mathcal{GCN}^*(\mathbf{x}_i) \quad (4)$$

we get $\Theta'_1 = \Theta_1 + \mathbf{I}$ where \mathbf{I} is identity matrix, and $\Theta'_2 = \Theta_2$ as weights of new fused graph convolution layer \mathcal{GCN}^* .

Fusing convolutional and batch normalization layers: As shown in Figure 5, there are three branches in a RepBlock. Among the branches, the single batch normalization (BN) layer can be equivalently transformed as a 1×1 convolutional layer whose convolution kernel is an identity matrix and a BN layer. Thus, in each branch, there is a convolutional layer with $n \times m$ kernel size (m can be equal to n) and a BN layer. For the convolutional layer, we use $\mathbf{W}^{n,m} \in \mathbb{R}^{C_2 \times C_1 \times n \times m}$ and b to represent its convolution kernel and bias. Here, C_2 and C_1 mean the number of output and input channels. For the BN layer, we use $\mu, \sigma, \gamma, \beta$ to represent its accumulated mean, standard deviation, learned scaling factor and bias, respectively. In regard to the input and output of the RepBlock module, they are represented with $\mathbf{x} \in \mathbb{R}^{N \times C_1 \times h_1 \times w_1}$ and $\mathbf{y} \in \mathbb{R}^{N \times C_2 \times h_2 \times w_2}$, respectively. Here, N is the batch size, while h_i and w_i , $i \in [1, 2]$ respectively mean the height and width of feature map. To fuse the original convolutional layer and BN layer, we adopt Equation 5 to get an alternative convolutional layer, which transforms the original two-step operation (i.e., convolution and BN) into one-step operation in a mathematically equivalent way.

$$\begin{aligned} \mathcal{BN}(\mathcal{CONV}(\mathbf{x})) &= \gamma \frac{\mathbf{W}^{n,m}(\mathbf{x}) + b - \mu}{\sigma} + \beta \\ &= \frac{\gamma}{\sigma} \mathbf{W}^{n,m}(\mathbf{x}) + \left(\frac{\gamma(b - \mu)}{\sigma} + \beta \right) \end{aligned} \quad (5)$$

In the new alternative convolutional layer, the convolution kernel is $\mathbf{W}^{n,m'} = \frac{\gamma}{\sigma} \mathbf{W}^{n,m}$, while the bias is $b' = \frac{\gamma(b - \mu)}{\sigma} + \beta$.

Merging multi-branches: In a RepBlock, when considering the kernel size difference in each branch, we adopt zero-padding to extend the 1×1 convolution kernel to $n \times n$ convolution kernel. Then, we use Equation 5 to transform each branch into a new $n \times n$ convolutional layer. After that, we use Equation 6 to add the three kernels $\mathbf{W}^{n'}$, $\mathbf{W}^{1'}$ and $\mathbf{W}^{0'}$ as well as the three bias vectors $b^{n'}$, $b^{1'}$, $b^{0'}$ to get the final convolution kernel and bias of the merged "RepBlock* $n \times n$ ", as shown in Figure 5 (e).

$$\begin{aligned} \mathbf{y} &= \mathcal{BN}(\mathbf{W}^n(\mathbf{x}) + b^n) + \mathcal{BN}(\mathbf{W}^1(\mathbf{x}) + b^1) + \mathcal{BN}(\mathbf{x}) \\ &= \mathbf{W}^{n'}(\mathbf{x}) + \mathbf{W}^{1'}(\mathbf{x}) + \mathbf{W}^{0'}(\mathbf{x}) + b^{n'} + b^{1'} + b^{0'} \\ &= (\mathbf{W}^{n'} + \mathbf{W}^{1'} + \mathbf{W}^{0'}) (\mathbf{x}) + (b^{n'} + b^{1'} + b^{0'}) \end{aligned} \quad (6)$$

3.4 Gloss Prediction and Sentence Generation

After getting the final feature representation of a sign language video, we design TextInfer module to predict glosses and generate the spoken sentence.

Gloss recognition: To predict the gloss sequence $G = \{g_t\}_{t=1}^v$, we adopt CTC loss to calculate the maximum probability of aligning input frames and output glosses.

Sentence generation: With the predicted gloss sequence $G = \{g_t\}_{t=1}^v$, we design a tiny translation network to generate the spoken language sentence $Y = \{w_t\}_{t=1}^r$. In the translation from gloss sequence to spoken language, word morphology changes due to restrictions on grammar rules, e.g., 'Lieb' in gloss sequence changes to 'Liebe' in spoken language, as shown in Figure 2. Thus we design a char-level encoder and a word-level encoder to learn correlations of the same word in different forms, aiming to benefit SLT using variants of a word.

Char-level encoder: For a gloss g_t , we use $\{c_t^j\}_{j=1}^{n_t}$ to represent its n_t characters. The initial embedding of char c_t^j is set as e_t^j , then we adopt a character-level encoder, which utilizes 1D convolution and max-pooling $\mathcal{MP}[\cdot]$ to get the fixed-length char-level embedding e_t of gloss g_t , as shown in Equation 7, where $[\cdot]$ is the concatenation operation.

$$e_t = \mathcal{MP}[\mathcal{CONV}_c(e_t^1), \mathcal{CONV}_c(e_t^2), \dots, \mathcal{CONV}_c(e_t^{n_t})] \quad (7)$$

Word-level encoder: We set the initial embedding of gloss g_t as r_t , then we adopt a word encoder, which combines the initial word-level embedding r_t and the char-level embedding e_t to get the word-level embedding r_t^* of gloss g_t , as follows.

$$r_t^* = [r_t, \mathcal{CONV}_w[r_t, e_t]] \quad (8)$$

Transformer decoder: With the gloss embeddings $\{r_t^*\}_{t=1}^v$, we use three-layered transformer with hidden size 512 and multi-head number 8 to predict the final spoken sentence. The transformer outputs the prediction probability of the word w_t step by step with "[SOS]" as the start symbol and "[EOS]" as the end symbol.

3.5 Multi-stage Training

We design a progressive optimization strategy to train our network step by step for better performance.

1) *Heatmap regression for keypoint detection:* The keypoint detection loss \mathcal{L}_H is calculated with Equation 9, where $M^H, M^G \in \mathbb{R}^{K \times h \times w}$ denote the predicted heatmap and the groundtruth heatmap, respectively, and K, h, w are the number of keypoints, the height and width of a heatmap. In our model, $K=3$ which represents nose, left/right wrists. In regard to the ground-truth heatmap, it has a heating area containing the keypoint, where the heating area is produced by applying a 2D Gaussian function with 1-pixel standard deviation [43] on the estimated keypoint from OpenPose [7].

$$\mathcal{L}_H = \frac{1}{K} \sum_k \sum_i^h \sum_j^w (M_{k,i,j}^H - M_{k,i,j}^G)^2 \quad (9)$$

2) *Joint loss for SLR:* We employ CTC loss to predict the gloss sequence G based on features \mathbf{O} . The CTC loss calculates the probability of all possible alignment paths Γ between \mathbf{O} and G , thus it is formalized as: $\mathcal{L}_{CTC} = -\ln(\sum_{\varphi \in \Gamma} p(\varphi|\mathbf{O}))$. Furthermore, to achieve the goal of SLR, we calculate the joint loss \mathcal{L}_{slr} based on \mathcal{L}_H and \mathcal{L}_{CTC} , as follows.

$$\mathcal{L}_{slr} = \mathcal{L}_H + \mathcal{L}_{CTC} \quad (10)$$

3) *Data augmentation for translation:* Finally, the cross entropy loss $\mathcal{L}_{slt} = -\sum_{t=1}^T \sum_{i=0}^V p(w_t^i) \log(p(\hat{w}_t^i))$ is adopted to train our

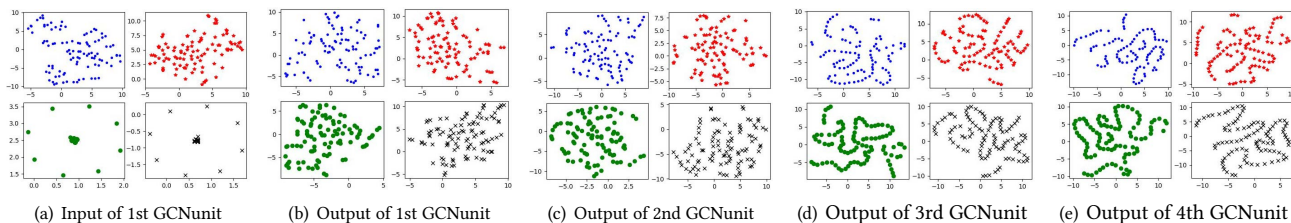


Figure 6: Visualization of feature distribution extracted by each GCN unit on one PHOENIX14T sample, which is displayed via t-SNE [44]. Blue, red, green and black points represent the feature of full frame, face, left hand and right hand, respectively.

translation network, where $p(\hat{w}_t)$ is the probability of predicted word and $p(w_t)$ is the probability of ground-truth word at the t th time step. In addition, considering that the limited number of gloss-sentence pairs (G_i, Y_i) provided by datasets hinders translation performance, we further introduce data augmentation for the translation network. Specifically, we take our SLR module as a generator network and add new gloss-text pairs (G'_i, Y_i) into training samples if $WER(G'_i, G_i) \leq 0.25$. The extended spoken language corpus reaches 100 times as large as the original corpus.

4 PERFORMANCE EVALUATION

4.1 Datasets

Three public datasets are used to evaluate our model in two aspects, i.e., SLR and SLT. (1) **CSL** [22] is a Chinese Sign Language dataset that contains 25K labeled videos with 100 Chinese sentences from 50 signers and it is used for CSLR in our experiment. (2) **PHOENIX14** [26] is a German Sign Language dataset that has been widely used for CSLR tasks. It contains 5672, 540, 629 weather forecasts samples from 9 signers for training, validation, and testing respectively. (3) **PHOENIX14T** [4] contains 7096, 519 and 642 samples for training, validation, and testing respectively. It has two-stage annotations: gloss annotations with a vocabulary of 1066 different signs for CSLR and German translation annotations with a vocabulary of 2877 different words for SLT.

4.2 Experimental Setting

Now, we describe the detailed setting in RTG-Net. *For data preprocessing*, each frame is reshaped into 224×224 pixels and the Gaussian noises are added for data augmentation. *For key region cropping*, the size of cropped face region is set to 30×30 , 40×40 and 40×40 pixels for CSL, PHOENIX14 and PHOENIX14T datasets, respectively. The size of cropped left or right hand region is set to 40×40 , 60×60 and 60×60 pixels for CSL, PHOENIX14 and PHOENIX14T datasets, respectively. *For model training*, the batch size is set to 24. Adam optimizer is used to optimize model parameters and initial learning rate is set to 0.0001 with a decay factor of 0.8. Besides, dropout with a rate of 0.5 is used after word embedding layer. *For model implementation*, our model is implemented with PyTorch 1.8 and trained for 100 epochs on 4 NVIDIA Tesla V100 GPUs. Besides we evaluate our model on an edge device without GPUs, i.e., Microsoft Surface Pro 6 configured with 1.90GHZ i7-8650U CPU and 16G RAM.

In regard to the performance metrics, we adopt the commonly-used Word Error Rate (WER) [26] as the metric for SLR tasks, while

Table 2: Ablation study of backbone network on PHOENIX14T. Edge(200) means the inference time of 200 frames on Microsoft Surface Pro 6

PHOENIX14T	DEV WER	TEST WER	Time(s) Edge(200)	Model		
				Size	FLOPs	Params
VGG	35.14	36.87	47.21	836.0	1570.9	217.04
Resnet18	19.42	19.75	26.58	203.0	782.2	51.53
MobileNetV1	34.59	33.45	11.56	81.4	153.5	19.57
MobileNetV2	32.46	36.55	12.75	65.5	64.1	15.97
RTG-Net	19.63	20.01	6.27	80.9	244.5	19.88

adopting the ROUGE-L F1-Score [33] and BLEU-1,2,3,4 [39] as the metrics for SLT tasks. In addition, to measure the resource and computation overhead of an approach, we also introduce the following metrics, i.e., model size (MB), the number of floating-point operations (FLOPs) (G), the quantity of parameters (M) and inference time (Seconds), and these metrics are measured on the same edge device (i.e., Microsoft Surface Pro 6).

4.3 Model Performance

To verify the effectiveness of proposed network (RTG-Net), we perform the ablation study and time analysis.

Ablation study: To estimate the contributions of designed components in RTG-Net, we perform the ablation study on PHOENIX14T dataset. In regard to the designed components, they mainly include feature extraction component, graph-related components, structural re-parameterization and data augmentation. In the ablation experiments, we remove only one type of component at a time, and use ‘DEV’ and ‘TEST’ to represent the validation set and test set, respectively.

1) *Feature extraction component:* To extract efficient features from each region, we design four subnetworks, which have the similar architecture and all adopt RepVGG as the backbone. To verify the effectiveness of the adopted backbone network in feature extraction, we replace our backbone with other mainstream networks (i.e., VGG, ResNet18, MobileNetV1, MobileNetV2), while keeping the dimension of output features unchanged. According to Table 2, only ResNet18 can achieve a comparable performance with ours (i.e., 19.42% WER in dev set), while the inference time of ResNet18 is more than four times of ours. Therefore, the RepVGG with structural re-parameterization in our model can achieve the best balance of SLR performance and computation overhead, which can efficiently extract features.

2) *Graph-related Components:* The graph-related components include the constructed region-aware temporal graph and S-GCN.

Table 3: Ablation study of RTG and S-GCN on PHOENIX14T.

PHOENIX14T	DEV	TEST	Time(s) Edge(200)	Model		
	WER	WER		Size	FLOPs	Params
w/o Face	23.25	24.74	6.10	74.9	243.3	18.31
w/o lhand	27.20	28.68	6.00	71.3	240.8	17.35
w/o rhand	29.53	32.55	6.00	71.3	240.8	17.35
w/o frame	40.33	39.87	5.92	62.2	226.8	15.26
AllConnect	19.73	19.94	6.65	80.9	244.5	19.88
LSTM	45.39	46.33	10.54	713.2	261.4	186.44
BiLSTM	38.75	40.22	20.50	1831.1	289.9	479.74
Transformer	28.92	29.22	17.86	971.2	270.8	62.75
ST-GCN	22.23	23.14	15.65	125.0	276.5	31.61
RTG-Net	19.63	20.01	6.27	80.9	244.5	19.88

Table 4: Ablation study of different number of GCN units on PHOENIX14T.

Num	DEV	TEST	Time(s) Edge(200)	Model		
	WER	WER		Size	FLOPs	Params
2	26.55	27.28	5.73	75.7	242.6	18.27
4	19.63	20.01	6.27	80.9	244.5	19.88
6	19.37	19.85	6.95	87.3	246.0	20.90
8	20.65	20.01	7.84	94.8	247.4	22.21
10	23.46	25.83	8.73	102.0	248.9	23.52

In regard to the constructed RTG, it is consisted of four kinds of nodes and edges. In the experiment, everytime we only remove one kind of node and the edge connected to these nodes. When removing the full-frame node, all the original edges will be removed, thus the remaining three nodes in a frame will be connected as a complete graph. According to Table 3, when removing the full frame, face, left or right hand region, the WER increases by 20.7%, 3.62%, 7.57% or 9.9% in dev set, respectively. It indicates that each region contributes to a better performance, especially the full frame which contains the global information of signs. In addition, we also verify the effectiveness of edges (i.e., connection ways between nodes). Specifically, instead of connecting each key region with the full frame, we connect every two regions to get a complete graph of a frame, which is possible to provide more interactions between nodes. According to Table 3, when using the complete graph ('AllConnect'), the SLR performance is quite close to that of our model, but the inference time is longer by 0.38s. Therefore, the good SLR performance and shorter inference time demonstrate the efficiency of designed RTG.

To verify the effectiveness of S-GCN, we first show the effect of different number of GCN units. As shown in Table 4, as the number of GCN units increases, the inference time increases while the SLR performance first increases and then decreases. The reason may be that the over-smoothing phenomenon in deep GCNs [32, 37] will decrease the distinctiveness of node features, which leads to performance degradation of SLR. According to Table 4, the model achieves the lowest WER (i.e., 19.37% WER on dev set) with 6 GCN units, and achieves the 19.63% WER with 4 GCN units. But the model takes 6.95s to process 200 frames (28.8 fps) with 6 GCN units while taking 6.27s (31.9 fps, i.e., larger than the default frame rate 30 fps and satisfying real-time requirement) to process 200 frames with 4 GCN units. Thus, we design our S-GCN with 4 GCN units

Table 5: Ablation study of structural re-parameterization on PHOENIX14T dataset. Note: 'w/o': without, 'SR': structural re-parameterization.

PHOENIX14T	DEV	TEST	Time(s) Edge(200)	Model		
	WER	WER		Size	FLOPs	Params
w/o SR	19.63	20.01	12.04	88.1	266.9	21.65
RTG-Net	19.63	20.01	6.27	80.9	244.5	19.88

Table 6: SLT performance when using different training data. 'R' represents ROUGE, and 'B1-4' denotes BLEU-1 to BLEU-4.

PHOENIX14T	DEV			TEST		
	R	B1	B4	R	B1	B4
Labeled gloss	44.31	45.85	18.90	43.19	44.53	19.91
Predicted gloss	47.61	48.53	22.69	47.34	46.84	22.75
Labeled+Predicted	50.18	51.17	25.95	50.04	50.87	25.87

to achieve the best balance of SLR performance and computation overhead.

In addition, we replace all ResGraphConv blocks in S-GCN module with mainstream temporal model, i.e., LSTM, BiLSTM and transform layer [45], while replacing each GCN unit with ST-GCN unit [49]. Specifically, we concatenate the features from four nodes as the input for LSTM, BiLSTM and transform layer, while keeping the dimension of the hidden layer unchanged. According to Table 3, when using alternative LSTM, BiLSTM, transform layer, and ST-GCN units, the WER increases by 26.32%, 20.21.1%, 9.21% and 3.13% in test set, respectively. Besides, the inference time by adopting LSTM, BiLSTM, transformer layer, and ST-GCN units increases by 4.27, 14.23, 11.59 and 9.38 seconds respectively, when compared with S-GCN. It indicates that the designed S-GCN is essential for real-time sign language processing on edge devices and contributes to a better performance with lower computation overhead.

To verify the effectiveness of both RTG and S-GCN on feature representation, we randomly select one sample in PHOENIX14T test set and visualize the feature distribution of S-GCN by t-SNE [44], as shown in Figure 6. In Figure 6(a), the initial feature distribution of consecutive frames in one video shows discontinuity, indistinguishability and irregularity, while S-GCN can distinguish features of each frame as shown in Figure 6(b) and 6(c). Furthermore, as features pass through deeper GCN units, feature distributions in each node show continuity, distinguishability and unique pattern, as shown in Figure 6(d) and 6(e), which indicates that designed RTG and S-GCN can learn the continuous changes, discriminative representations and regularity of signs in one video.

3) *Effect of structural re-parameterization:* We show the ablation study on structural re-parameterization in Table 5. The results show that after applying structural re-parameterization, the designed RepBlocks (shown in Section 3.3) can efficiently reduce inference time. Besides, the performance of simplified model is equivalent to that of the original model.

4) *Effect of data augmentation on translation:* In Table 6, we show the performance when using different training data. When adding predicted gloss sequence into training corpus ('Labeled+Predicted'), the ROUGE score increases from 44.31% to 50.18% on test set, compared with the method only using the labeled training samples provided by dataset ('Labeled gloss'), which demonstrates that the added gloss-sentence pairs can improve model performance.

Table 7: Comparison on PHOENIX14T dataset. Note: OM: ‘OUT OF MEMORY’. Edge(200)/GPU(200) means the inference time of 200 frames on Microsoft Surface Pro 6/Tesla V100 GPU. 200 is the max length of a video.

PHOENIX14T	DEV					TEST					Time(s)		Model		
	R	B1	B2	B3	B4	R	B1	B2	B3	B4	Edge(200)	GPU(200)	Size	FLOPs	Params
TSPNet [30]	-	-	-	-	-	34.96	36.10	23.12	16.68	13.41	OM	-	559.0	2250.9	70.98
H+M+P [5]	45.90	-	-	-	19.50	43.60	-	-	-	18.30	-	-	-	-	-
S2G2T [4]	44.14	42.88	30.30	23.02	18.40	43.80	43.29	30.29	22.82	18.13	43.38	0.16	1658.8	336.8	100.01
(G)S2(G+T) [6]	-	47.26	34.40	27.05	22.38	-	46.61	33.73	26.19	21.31	OM	0.17	159.7	152.3	31.24
DeepHand [38]	-	-	-	-	-	38.05	38.50	25.64	18.59	14.56	49.15	0.16	290.0	167.6	91.18
STMC-T [57]	48.24	47.60	36.43	29.18	24.09	46.65	46.98	36.09	28.70	23.65	130.2	0.39	435.0	1578.2	111.13
HST-GNN [23]	-	46.10	33.40	27.50	22.60	-	45.20	34.70	27.10	22.30	-	-	-	-	-
SimuSLT [50]	49.21	47.76	35.33	27.85	22.85	49.23	48.23	35.59	28.04	23.14	62.69	0.72	390.0	616.4	41.24
STMC-Tran [52]	48.70	48.27	35.20	27.47	22.47	46.77	48.73	36.53	29.03	24.00	OM	0.40	435.0	1579.3	136.33
BN-TIN [55]	50.29	51.11	37.90	29.80	25.94	49.54	50.80	37.75	29.72	24.32	94.67	0.86	598.0	532.3	117.82
RTG-Net	50.18	51.17	37.95	29.88	25.95	50.04	50.87	37.95	29.74	25.87	6.33	0.15	103.0	244.9	24.75

Table 8: Comparison on PHOENIX14 dataset.

PHOENIX14	DEV		TEST		Time(s)		Model		
	WER	Del/Ins	WER	Del/Ins	Edge(200)	GPU(200)	Size	FLOPs	Params
Align [42]	37.1	12.9/2.6	36.7	13.0/2.5	28.44	0.25	668.0	25.3	167.93
SBD-RL [48]	28.6	9.9/5.6	28.6	8.9/5.1	71.75	0.11	219.0	2332.6	57.43
Re-sign [29]	27.1	-/-	26.8	-/-	48.87	0.19	219.0	309.9	51.15
DNF [12]	23.1	7.3/3.3	22.9	6.7/3.3	69.50	0.16	309.0	367.5	58.91
FCN [10]	23.7	-/-	23.9	-/-	-	-	-	-	-
Boosting [40]	21.3	7.3/2.7	21.9	7.3/2.4	19.63	0.19	685.0	499.5	173.24
STMC-R [56]	21.1	7.7/3.4	20.7	7.4/2.6	121.40	0.16	252.0	1573.8	66.25
RTG-Net	20.0	8.4/1.5	20.1	8.6/1.7	6.27	0.11	80.9	244.5	19.88

Inference time analysis: We evaluate the inference time of a sign language video with 300 frames on CSL and 200 frames on both PHOENIX14 and PHOENIX14T datasets on Surface Pro6, and then average the time of 100 runs as the reported inference time. As shown in Table 2 and Table 7, RTG-Net takes 6.27 seconds to get the recognized gloss sequence and takes 6.33 seconds to get the translated sentence for 200 frames. The experimental results prove that the processing speed of RTG-Net for SLT achieves 30 frames per second (fps) and satisfies the real-time requirement on edge devices, i.e., the processing speed of RTG-Net is larger than the frame rate.

4.4 Comparison

We also compare RTG-Net with existing approaches in SLR or SLT. Here, the recognition/translation performance was reported by the existing work itself. While for resource and computation overhead (i.e., model size, FLOPs, quantity of parameters, inference time), they are provided by our re-implemented model for reference, since those information was usually not provided by the existing work.

Evaluation on PHOENIX14T dataset: According to Table 7, our RTG-Net can achieve the state-of-the-art performance compared with the existing models. Specifically, the best BLEU1 (i.e., 50.87%) is achieved by our RTG-Net on the test set. Besides, our RTG-Net can achieve the lowest inference time (i.e., 6.33s), which is only about one twentieth of that of STMC-T. Consequently, only RTG-Net can balance the translation performance and computation overhead well, and satisfies the real-time requirement.

Evaluation on PHOENIX14 dataset: As shown in Table 8, our RTG-Net can achieve the comparable performance with the existing approaches on PHOENIX14 dataset for SLR. Specifically, on the ‘TEST’ set, the lowest WER 20.1% is achieved by our RTG-Net. Besides, when moving to the inference time, the time of STMC-R is

Table 9: Comparison on CSL dataset.

CSL	SPLIT I		SPLIT II		Time(s)		Model		
	WER	WER	Edge(300)	GPU(300)	Size	FLOPs	Params		
PyramidBi [31]	9.1	59.4	-	-	-	-	-	-	
LSHAN [22]	17.3	-	OM	OM	1747.6	6179.6	458.13		
STMC-R [56]	2.1	28.6	OM	OM	252.0	1573.8	66.24		
Align [42]	-	32.7	39.4	0.35	668.0	1303.3	167.93		
SBD-RL [48]	-	26.8	OM	-	219.0	2332.6	57.43		
RTG-Net	0.8	23.2	9.19	0.16	72.7	204.7	17.73		

much larger than (i.e., more than 20 times) that of our RTG-Net, and only our RTG-Net can achieve real-time sign language processing on edge devices (i.e., processing speed is larger than 30 fps).

Evaluation on CSL dataset: Following previous work [17], we compare our network with existing approaches on two settings: *Split I- signer independent test* and *Split II-unseen sentences test*. As shown in Table 9, our RTG-Net achieves the best performance, i.e., 0.8% WER on Split I and 23.2% WER on Split II. Besides, we can also achieve the smallest inference time (i.e., 9.19s), which indicates that our RTG-Net can achieve SOTA performance with limited computation overhead.

5 CONCLUSION

In this paper, we propose a RTG-Net for real-time video-based SLR and SLT on edge devices. Specifically, to achieve the real-time requirement, we design a shallow graph convolutional network and adopt structural re-parameterization to reduce computation overhead. To guarantee the SLR or SLT performance, we extract key regions, and then construct a region-aware temporal graph, to improve the feature representation of a sign language video. Extensive experimental results demonstrate that RTG-Net can achieve real time SLR or SLT on edge devices, while having a comparable performance with the existing approaches.

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, 61832008, 61872174. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

- [1] Kshitij Bantupalli and Ying Xie. 2018. American sign language recognition using deep learning and computer vision. In *2018 Big Data*. IEEE, 4896–4899.
- [2] Jan Bungeroth and Hermann Ney. 2004. Statistical sign language translation. In *Workshop on representation and processing of sign languages, LREC*, Vol. 4. Citeseer, 105–108.
- [3] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, and Richard Bowden. 2017. Subnets: End-to-end hand shape and continuous sign language recognition. In *ICCV*. IEEE, 3075–3084.
- [4] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. 2018. Neural Sign Language Translation. In *CVPR*. 7784–7793. <https://doi.org/10.1109/CVPR.2018.00812>
- [5] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. 2020. Multi-channel Transformers for Multi-articulatory Sign Language Translation. *arXiv preprint arXiv:2009.00299* (2020).
- [6] Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. 2020. Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation. In *CVPR*. 10023–10033.
- [7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*. 7291–7299.
- [8] Yutong Chen, Fangyun Wei, Xiao Sun, Zhirong Wu, and Stephen Lin. 2022. A simple multi-modality transfer learning baseline for sign language translation. In *CVPR*. 5120–5130.
- [9] Yutong Chen, Ronglai Zuo, Fangyun Wei, Yu Wu, Shujie Liu, and Brian Mak. 2022. Two-stream network for sign language recognition and translation. *Advances in Neural Information Processing Systems* 35 (2022), 17043–17056.
- [10] Ka Leong Cheng, Zhaoyang Yang, Qifeng Chen, and Yu-Wing Tai. 2020. Fully convolutional networks for continuous sign language recognition. In *European Conference on Computer Vision*. Springer, 697–714.
- [11] Rumpeng Cui, Hu Liu, and Changshui Zhang. 2017. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7361–7369.
- [12] Rumpeng Cui, Hu Liu, and Changshui Zhang. 2019. A deep neural framework for continuous sign language recognition by iterative training. *MM* 21, 7 (2019), 1880–1891.
- [13] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. 2021. Revgg: Making vgg-style convnets great again. In *CVPR*. 13733–13742.
- [14] Shiwei Gan, Yafeng Yin, Zhiwei Jiang, Lei Xie, and Sanglu Lu. 2021. Skeleton-Aware Neural Sign Language Translation. In *ACM MM*. 4353–4361.
- [15] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*. 369–376.
- [16] K. Grobel and M. Assan. 1997. Isolated sign language recognition using hidden Markov models. In *SMC*, Vol. 1. 162–167 vol.1. <https://doi.org/10.1109/ICSMC.1997.625742>
- [17] Dan Guo, Wengang Zhou, Anyang Li, Houqiang Li, and Meng Wang. 2019. Hierarchical recurrent deep fusion using adaptive clip summarization for sign language translation. *TIP* 29 (2019), 1575–1590.
- [18] Dan Guo, Wengang Zhou, Meng Wang, and Houqiang Li. 2016. Sign language recognition based on adaptive hmms with data augmentation. In *ICIP*. IEEE, 2876–2880.
- [19] Aiming Hao, Yuecong Min, and Xilin Chen. 2021. Self-Mutual Distillation Learning for Continuous Sign Language Recognition. In *ICCV*. 11303–11312.
- [20] Hezhen Hu, Wengang Zhou, and Houqiang Li. 2021. Hand-Model-Aware Sign Language Recognition. In *AAAI*, Vol. 35. 1558–1566.
- [21] Jie Huang, Wengang Zhou, Houqiang Li, and Weiping Li. 2018. Attention-based 3D-CNNs for large-vocabulary sign language recognition. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 9 (2018), 2822–2832.
- [22] Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, and Weiping Li. 2018. Video-based sign language recognition without temporal segmentation. In *AAAI*.
- [23] Jichao Kan, Kun Hu, Markus Hagenbuchner, Ah Chung Tsoi, Mohammed Benamoun, and Zhiyong Wang. 2022. Sign Language Translation with Hierarchical Spatio-Temporal Graph Neural Network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3367–3376.
- [24] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [25] Oscar Koller, Cihan Camgoz, Hermann Ney, and Richard Bowden. 2019. Weakly supervised learning with multi-stream CNN-LSTM-HMMs to discover sequential parallelism in sign language videos. *TPAMI* (2019).
- [26] Oscar Koller, Jens Forster, and Hermann Ney. 2015. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *CVIU* 141 (Dec. 2015), 108–125.
- [27] Oscar Koller, Hermann Ney, and Richard Bowden. 2016. Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In *CVPR*. 3793–3802.
- [28] Oscar Koller, O Zargaran, Hermann Ney, and Richard Bowden. 2016. Deep sign: Hybrid CNN-HMM for continuous sign language recognition. In *BMVC*.
- [29] Oscar Koller, Sepehr Zargaran, and Hermann Ney. 2017. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs. In *CVPR*. 4297–4305.
- [30] Dongxu Li, Chenchen Xu, Xin Yu, Kaihao Zhang, Benjamin Swift, Hanna Suominen, and Hongdong Li. 2020. TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation. In *NIPS*, Vol. 33.
- [31] Haibo Li, Liqing Gao, Ruize Han, Liang Wan, and Wei Feng. 2020. Key action and joint ctc-attention based sign language recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2348–2352.
- [32] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.
- [33] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.
- [34] Tao Liu, Wengang Zhou, and Houqiang Li. 2016. Sign language recognition with long short-term memory. In *ICIP*. IEEE, 2871–2875.
- [35] Yuecong Min, Aiming Hao, Xiujuan Chai, and Xilin Chen. 2021. Visual alignment constraint for continuous sign language recognition. In *ICCV*. 11542–11551.
- [36] Zhe Niu and Brian Mak. 2020. Stochastic fine-grained labeling of multi-state sign glosses for continuous sign language recognition. In *European Conference on Computer Vision*. Springer, 172–186.
- [37] Kenta Oono and Taiji Suzuki. 2019. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947* (2019).
- [38] Alptekin Orbay and Lale Akarun. 2020. Neural sign language translation by learning tokenization. In *FG 2020*. IEEE, 222–228.
- [39] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*. 311–318.
- [40] Junfu Pu, Wengang Zhou, Hezhen Hu, and Houqiang Li. 2020. Boosting Continuous Sign Language Recognition via Cross Modality Augmentation. In *MM*. 1497–1505.
- [41] Junfu Pu, Wengang Zhou, and Houqiang Li. 2018. Dilated Convolutional Network with Iterative Optimization for Continuous Sign Language Recognition.. In *IJCAI*, Vol. 3. 7.
- [42] Junfu Pu, Wengang Zhou, and Houqiang Li. 2019. Iterative alignment network for continuous sign language recognition. In *CVPR*. 4165–4174.
- [43] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep high-resolution representation learning for human pose estimation. In *CVPR*. 5693–5703.
- [44] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
- [45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
- [46] Andreas Veit, Michael J Wilber, and Serge Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. *NIPS* 29 (2016), 550–558.
- [47] Hanjie Wang, Xiujuan Chai, Xiaopeng Hong, Guoying Zhao, and Xilin Chen. 2016. Isolated sign language recognition with grassmann covariance matrices. *TACCESS* 8, 4 (2016), 1–21.
- [48] Chengcheng Wei, Jian Zhao, Wengang Zhou, and Houqiang Li. 2020. Semantic Boundary Detection with Reinforcement Learning for Continuous Sign Language Recognition. *TCSVT* 31, 3 (2020), 1138–1149.
- [49] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*.
- [50] Aoxiong Yin, Zhou Zhao, Jinglin Liu, Weike Jin, Meng Zhang, Xingshan Zeng, and Xiaofei He. 2021. SimulSLT: End-to-End Simultaneous Sign Language Translation. In *MM*. 4118–4127.
- [51] Kayo Yin, Amit Moryossef, Julie Hochgesang, Yoav Goldberg, and Malihe Alikhani. 2021. Including signed languages in natural language processing. *arXiv preprint arXiv:2105.05222* (2021).
- [52] Kayo Yin and Jesse Read. 2020. Better sign language translation with STMC-transformer. In *COLING*. 5975–5989.
- [53] Jihai Zhang, Wengang Zhou, and Houqiang Li. 2014. A threshold-based hmm-dtw approach for continuous sign language recognition. In *ICIMCS*. 237–240.
- [54] Hao Zhou, Wengang Zhou, and Houqiang Li. 2019. Dynamic pseudo label decoding for continuous sign language recognition. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1282–1287.
- [55] Hao Zhou, Wengang Zhou, Weizhen Qi, Junfu Pu, and Houqiang Li. 2021. Improving Sign Language Translation with Monolingual Data by Sign Back-Translation. In *CVPR*. 1316–1325.
- [56] Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. 2020. Spatial-Temporal Multi-Cue Network for Continuous Sign Language Recognition. In *AAAI*. 13009–13016.
- [57] Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. 2021. Spatial-temporal multi-cue network for sign language recognition and translation. *TMC* (2021).
- [58] Ronglai Zuo and Brian Mak. 2022. C2SLR: Consistency-Enhanced Continuous Sign Language Recognition. In *CVPR*. 5131–5140.

A APPENDIX

A.1 The detailed information of CSL, PHOENIX14 and PHOENIX14T

Here, we provide more detailed information about the datasets. For CSL, the frame rate is 30 fps and the frame’s resolution is 1280×720 pixels, we first crop each frame around its center to get a region with 500×500 pixels, then resize the cropped region to 224×224 pixels. In regard to the detailed settings for Split I and Split II in CSL, we follow the same settings with previous work [17]. (a) Split I- signer independent test: we select the sign language videos generated by 40 signers and that generated by the other 10 signers as the training set and test set, respectively. The training and testing sets have

Table 10: Comparison of SLR performance on PHOENIX14 dataset

Model	DEV		TEST	
	WER	del/ins	WER	del/ins
DeepHand [27]	47.1	16.3/4.6	45.1	15.2/4.6
SubUNet [3]	40.8	14.6/4.0	40.7	14.3/4.0
Staged-Opt [11]	39.4	13.7/7.3	38.7	12.2/7.5
Re-Sign [29]	27.1	-/-	26.8	-/-
DCN [41]	38.0	8.3/4.8	37.3	7.6/4.8
Weakly [25]	26.0	-/-	26.0	-/-
DNF(Flow) [12]	23.1	7.3/3.3	22.9	6.7/3.3
DPD+TEM [54]	35.6	9.5/3.2	34.5	9.3/3.1
Align-iOpt [42]	37.1	12.6/2.6	36.7	13.0/2.5
CMA [40]	21.3	7.3/2.7	21.9	7.3/2.4
SFL [36]	24.9	10.3/4.1	25.3	10.4/3.6
SBD-DL [48]	28.6	9.9/5.6	28.6	8.9/5.1
STMC [56]	21.1	7.7/3.4	20.7	7.4/2.6
SMKD [19]	20.8	6.8/2.5	21.0	6.3/2.3
FCN [10]	23.7	-/-	23.9	-/-
VAC [35]	21.2	7.9/2.5	22.3	8.4/2.6
C2SLR [58]	20.5	-/-	20.4	-/-
RTG-Net	20.0	8.4/1.5	20.1	8.6/1.7

Table 11: Comparison of SLR performance on PHOENIX14T dataset.

Model	DEV	TEST
	WER	WER
Weakly [25]	22.1	24.1
STMC [56]	19.6	21.0
SFL [36]	25.1	26.1
SLT [6]	24.6	24.5
FCN [10]	23.3	25.1
SMKD [19]	20.8	22.4
C2SLR [58]	20.2	20.4
RTG-Net	19.6	20.0

the same sentences with no overlap of signers. (b) Split II-unseen sentences test: we select the sign language videos corresponding to 94 sentences as the training set and the videos corresponding to the remaining 6 sentences as the test set. The signers and vocabulary of the training set and test set are the same, while the sentences have no overlaps. In regard to PHOENIX14 and PHOENIX14T, the frame rate is 25fps, the frame’s resolution is 210×260 pixels, and each frame is resized to 224×224 pixels.

A.2 Comparison

Evaluation of SLR performance on PHOENIX14 dataset: As shown in Table 10, we show the comparison results with existing methods on PHOENIX14 dataset. Compared with Table 8 in the main paper, results in this table contain extra models that have not been re-implemented, thus we don’t show metrics like: model size (MB), FLOPs (G), the quantity of parameters (M) and inference time (Seconds) here. According to Table 10, our RTG-Net can achieve the state-of-the-art performance on SLR task.

Evaluation of SLR performance on PHOENIX14T dataset: In Table 7 of the main paper, we only show the comparison of SLT results on PHOENIX14T dataset. Here, we show the comparison of SLR results. As shown in Table 11, our RTG-Net can achieve comparable performance with the existing approaches.

Evaluation of SLR and SLT performance on CSL-daily dataset: There is another sign language dataset that can be used for both SLR and SLT, called CSL-daily [55]. The CSL-daily dataset contains 18401, 1077 and 1176 labeled videos for training, validation and testing, and it has both gloss annotations with a vocabulary of 2000 different signs and Chinese translation annotations with a vocabulary of 2343 words.

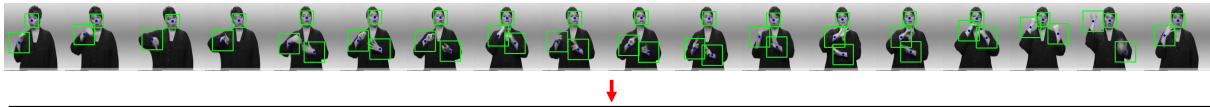
There is a little difference between the official dataset [55] and our received dataset, as shown in Table 12. Still, as shown in Table 13 and Table 14, our RTG-Net can outperform the state-of-the-art SLR models and achieves a comparable performance with the state-of-the-art SLT methods on CSL-daily.

Table 12: Difference between the official CSL-daily dataset and our received dataset.

Model	Official			Ours		
	Train	Dev	Test	Train	DEV	TEST
frames	2,227,178	124,530	153,074	2,185,328	131,931	150,248
resolution	1920×1080	← same		512 × 512	← same	
type	video	← same		images	← same	
FPS	30	← same		None	← same	

Table 13: Comparison of SLR performance on CSL-daily.

Model	DEV		TEST	
	WER	del/ins	WER	del/ins
SL-Transf [6]	33.1	10.3/4.4	32.0	9.6/4.1
BN-TIN [55]	33.6	13.9/3.4	33.1	13.5/3.0
RTG-Net	30.5	12.8/3.5	30.4	13.3/2.3



SLR	w/o Face:	donnerstag	ist	es	suedost	weiter	wechselhaft	nordwest	mehr	**	sonne
	w/o Lhand:	donnerstag	ist	**	suedost	weiter	wechselhaft	nordwest	mehr	freundlich	**
	w/o Rhand:	donnerstag	**	es	suedost	weiter	wechselhaft	west	mehr	freundlich	sonne
	w/o Frame:	donnerstag	**	**	suedost	weiter	wechselhaft	nordwest	mehr	freundlich	**
	Allconnect:	donnerstag	**	es	suedost	weiter	wechselhaft	nordwest	mehr	freundlich	sonne
	RTG-Net:	donnerstag	**	es	suedost	weiter	wechselhaft	nordwest	mehr	freundlich	sonne
	GT:	donnerstag	ist	es	suedost	weiter	wechselhaft	nordwest	mehr	freundlich	sonne
SLT	Labeled:	am donnerstag	** **	und im	süosthälfte	im	wechselhaft	nach sonne	später	**	es freundlicher
	Predicted:	am donnerstag	ist es **	im	süosthälfte	**	wechselhaft	nach westen	**	ist es freundlich	
	L+P:	am donnerstag	ist es in	im	süosthälfte	weiterhin	wechselhaft	nach nordwesten	hin ist es	freundlicher	
	GT:	am donnerstag	ist es in	der	süosthälfte	weiterhin	wechselhaft	nach nordwesten	hin ist es	freundlich	

Figure 7: Qualitative analysis on PHOENIX14T dataset. Note ‘w/o’:without, ‘Labeled’: Labeled gloss, ‘Predicted’: Predicted gloss, ‘L+P’:Labeled gloss+Predicted gloss, ‘GT’:Ground Truth, ‘**’: blank token.

Table 14: Comparison of SLT performance with other models on CSL-daily.

Model	PHOENIX14T DEV					PHOENIX14T TEST				
	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4
SL-Luong [4]	40.18	41.46	25.71	16.57	11.06	40.05	41.55	25.73	16.54	11.03
SL-Transf [6]	44.18	46.82	32.22	22.49	15.94	44.81	47.09	32.49	22.61	16.24
BN-TIN [55]	49.49	51.46	37.23	27.51	20.80	49.31	51.42	37.26	27.76	21.34
RTG-Net	49.12	51.57	37.65	27.19	20.81	50.15	51.33	37.37	27.81	21.74

A.3 Qualitative Analysis

In Figure 7, we show an example of PHOENIX14T test set. The image sequence in top part shows that our model can efficiently extract keypoints and crop key regions, the gloss sequence in middle part shows that each designed component contributes to the higher SLR performance, and the word sequence in the bottom part shows that proposed data augmentation on gloss-sentence pairs in Section 3.5 can further improve SLT performance.

A.4 Limitation Analysis

Unsegmented sign video: Like the most existing models, our model is trained under benchmark datasets where sign sentence segmentation is done by datasets. For an unsegmented long video,

it is necessary to segment the long video in the sentence level, since that the most current models can hardly perform SLR and SLT on videos without sentence segmentation. However, the sentence segmentation remains an unsolved research problem. Thus, despite the good SLR, SLT performance provided by our model and other models, there are still many problems to be solved before technological readiness for real-life applications.

Re-implemented results: In Section 4.4 of the main paper, metrics of existing models like: model size, FLOPs, quantity of parameters and inference time are provided by our re-implemented models, considering the source codes of the current models are generally not publicly available. We have tried our best to re-implement their models according to their papers and make their results comparable to ours. Still, this information is for reference.