Skeleton-Aware Neural Sign Language Translation

Shiwei Gan, Yafeng Yin*, Zhiwei Jiang, Lei Xie, Sanglu Lu
State Key Laboratory for Novel Software Technology, Nanjing University, China
gsw@smail.nju.edu.cn,yafeng@nju.edu.cn,jzw@nju.edu.cn,lxie@nju.edu.cn,sanglu@nju.edu.cn

ABSTRACT

As an essential communication way for deaf-mutes, sign languages are expressed by human actions. To distinguish human actions for sign language understanding, the skeleton which contains position information of human pose can provide an important cue, since different actions usually correspond to different poses/skeletons. However, skeleton has not been fully studied for Sign Language Translation (SLT), especially for end-to-end SLT. Therefore, in this paper, we propose a novel end-to-end Skeleton-Aware neural Network (SANet) for video-based SLT. Specifically, to achieve end-to-end SLT, we design a self-contained branch for skeleton extraction. To efficiently guide the feature extraction from video with skeletons, we concatenate the skeleton channel and RGB channels of each frame for feature extraction. To distinguish the importance of clips, we construct a skeleton-based Graph Convolutional Network (GCN) for feature scaling, i.e., giving importance weight for each clip. The scaled features of each clip are then sent to a decoder module to generate spoken language. In our SANet, a joint training strategy is designed to optimize skeleton extraction and sign language translation jointly. Experimental results on two large scale SLT datasets demonstrate the effectiveness of our approach, which outperforms the state-of-the-art methods. Our code is available at https://github.com/SignLanguageCode/SANet.

CCS CONCEPTS

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

KEYWORDS

Sign Language Translation; Skeleton; Neural Network

ACM Reference Format:
Shiwei Gan, Yafeng Yin, Zhiwei Jiang, Lei Xie, Sanglu Lu. 2021. Skeleton-Aware Neural Sign Language Translation. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475577

*Yafeng Yin is the corresponding author.

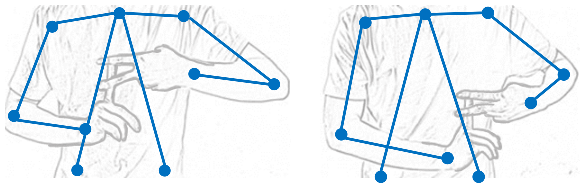

(a) Sign: doing (b) Sign: liver

Figure 1: In sign languages, the same hand gesture in different positions can have different meanings. The blue points and lines represent the skeleton.

1 INTRODUCTION

Sign language has been widely adopted as a communication way for deaf-mutes. To build the bridge between deaf-mutes and hearing people, the research work on sign language understanding emerged and the existing work can mainly be categorized as Sign Language Recognition (SLR) [21, 29, 45] and Sign Language Translation (SLT) [5, 15, 25]. Earlier, the sign language-related work usually focused on SLR, which aims at recognizing isolated sign as word or expression [12, 24, 37], or recognizing continuous signs as corresponding word sequence [4, 10, 13, 21]. However, the SLR work neglected the difference between sign language and spoken language on grammatical rules, i.e., the recognized word sequence may be not grammatically correct, thus hindering the understanding of sign language. Recently, due to the advancement of annotated dataset and deep learning technology, SLT has attracted people's attention. SLT is a more challenging task and its objective is to translate sign language to spoken language, while requiring that the translation results conform to the grammatical rules and linguistic characteristics of the target spoken language.

In regard to SLT, the prior work tended to decompose SLT into two stages, i.e., recognizing continuous signs as word sequence, and then utilizing language models to construct sentences with the words [3, 9]. However, the two-stage methods usually required gloss[1] annotation, which was a labor-intensive task and needed specialists. Recently, due to the development of deep learning technology, Camgoz et al. approached SLT as a neural machine translation task [5], and introduced the encoder-decoder network and attention mechanism for end-to-end SLT for the first time. After that, Camgoz et al. introduced transformer networks for end-to-end SLT from videos [7]. When considering the modality difference between video and language in SLT, feature representation was adopted, i.e., the video is represented as features which are later translated to language. However, in the existing neural-based methods, the feature representation of video was mainly consisted of full-frame [5, 15, 22] or local-area [6] features, while the skeleton information which reflects the important spatial structure of human

[1]Here, 'gloss' means a gesture with its closest meaning in natural languages [10].

pose in sign languages has not been fully studied. In fact, the skeleton can be used to distinguish signs with different human poses (i.e., different relative positions of hands, arms, etc), especially for the signs which use the same hand gesture in different positions to represent different meanings, as shown in Figure 1. Therefore, it is meaningful to introduce skeleton information into SLT.

To utilize skeletons for SLT, there emerged some work recently [6, 14], which advanced the research of skeleton assisted SLT. However, the existing research often had the following problems. First, obtaining the skeletons often required an external device [14] or extra offline preprocessing [6], which hindered the end-to-end SLT from videos. Second, the videos and skeletons were used as two independent data sources for feature extraction, i.e., not fused at the initial stage of feature extraction, thus the video-based feature extraction may be not efficiently guided/enhanced with the skeleton information. Third, each clip (i.e., a short segmented video) was usually treated equally, while neglecting the different importance of meaningful (e.g, sign-related) clips and unmeaninngful (e.g, end state) clips. Among the problems, the third one exists not only in skeleton-assisted SLT, but also in much SLT work.

To address the above three problems, we propose a Skeleton-Aware neural Network (SANet). Firstly, to achieve end-to-end SLT, SANet designs a self-contained branch for skeleton extraction. Secondly, to guide the video-based feature extraction with skeletons, SANet concatenates the skeleton channel and RGB channels for each frame, thus the features extracted from images/videos will be affected by skeleton. Thirdly, to distinguish the importance of clips, SANet constructs a skeleton-based Graph Convolutional Network (GCN) for feature scaling, i.e., giving importance weight for each clip. Specifically, SANet consists of four components, i.e., FrmSke, ClipRep, ClipScl, LangGen. At first, FrmSke is used to extract skeleton from each frame and frame-level features for a clip by convolutions and deconvolutions. Then, ClipRep is used to enhance clip representation by adding skeleton channel. After that, ClipScl is used to scale the clip representation by a skeleton-based Graph Convolutional Network (GCN). Finally, with the scaled features of clips, LangGen is used to generate spoken language with sequence to sequence learning. In addition, we design a joint optimization strategy for model training and achieve end-to-end SLT.

We make the following contributions in this paper.

- We propose a Skeleton-Aware neural Network (SANet) for end-to-end SLT, where a self-contained branch is designed for skeleton extraction and a joint training strategy is designed to optimize skeleton extraction and sign language translation jointly.
- We concatenate the extracted skeleton channel and RGB channels in source data level, thus can highlight human pose-related features and enhance the clip representation.
- We construct skeleton-based graphs and use graph convolutional network to scale the clip representation, i.e., weighting the importance of each clip, thus can highlight meaningful clips while weakening unmeaningful clips.
- We conduct extensive experiments on two large-scale public SLT datasets. The experimental results demonstrate that our SANet outperforms the state-of-the-art methods.

2 RELATED WORK

The existing research work on sign languages can be mainly categorized into SLR and SLT, where SLR can be further classified into isolated SLR and continuous SLR. In this section, we review the related work on isolated SLR, continuous SLR, and SLT.

Isolated Sign Language Recognition (ISLR): The isolated SLR aims at recognizing one sign as word or expression [2, 12, 24] which is similar to gesture recognition [27, 42] and action recognition [32, 40]. The early methods tended to select features from videos manually, and introduced Hidden Markov Model (HMM) [12, 16] to analyze the gesture sequence of a sign (i.e., human action) for recognition. However, the manually-selected features may limit the recognition performance. Therefore, in recent years, the deep learning-based approaches were introduced for isolated SLR. The approaches utilized neural networks to automatically extract features from videos [17]), Kinect's sensor data [36], or moving trajectories of skeleton joints [24] for isolated SLR, and often achieved a better performance.

Continuous Sign Language Recognition (CSLR): Continuous SLR aims at recognizing a sequence of signs to the corresponding word sequence [21, 28, 45], thus continuous SLR is more challenging than isolated SLR. To realize CSLR, the traditional methods like DTW-HMM [44] and CNN-HMM [21] introduced temporal segmentation and Hidden Markov Model (HMM) to transform continuous SLR to isolated SLR. Considering the possible errors and annotation burden in temporal segmentation, the recent deep learning based methods [18] applied sequence to sequence learning for continuous SLR. They learned the correspondence between two sequences from weakly annotated data in an end-to-end manner. However, many approaches tended to adopt Connectionist Temporal Classification (CTC) loss [13, 20, 43, 45] which requires that source and target sequences have the same order. In fact, the sign sequence in sign language and the word sequence in spoken language can be different [5], thus the approaches for continuous SLR are not suitable for SLT.

Sign Language Translation (SLT): SLT aims to translate sign languages into spoken languages. Traditional methods [3, 9] usually decomposed SLT into two stages, i.e., continuous SLR and text-to-text translation. The two-stage methods had both gloss annotations and sentence annotations, thus can be optimized in two stages for a better performance [5]. However, annotating glosses requires specialists and is a labor-intensive task. Recently, due to the advancement of public datasets in sentence-level annotations [5, 11, 18] and deep learning technology, there emerged a few end-to-end SLT approaches. Camgoz et al. [5] introduced the encoder-decoder framework to realize end-to-end SLT. Guo et al. [14, 15] proposed the hierarchical-LSTM model for end-to-end SLT. Camgoz et al. utilized the transformer networks [38] to jointly solve SLR and SLT[7]. Li et al. developed a temporal semantic pyramid encoder and a transformer decoder for SLT [22]. These neural-based approaches often adopted encoder and decoder for SLT.

To represent the sign languages, the existing neural-based SLT methods mainly focused on extracting full-frame [5, 15, 22] or local-area features [6, 46] from the video. There was only a little work paying attention on skeleton information (i.e., human pose) for SLT. Specifically, HRF [14] collected skeletons with a depth camera,

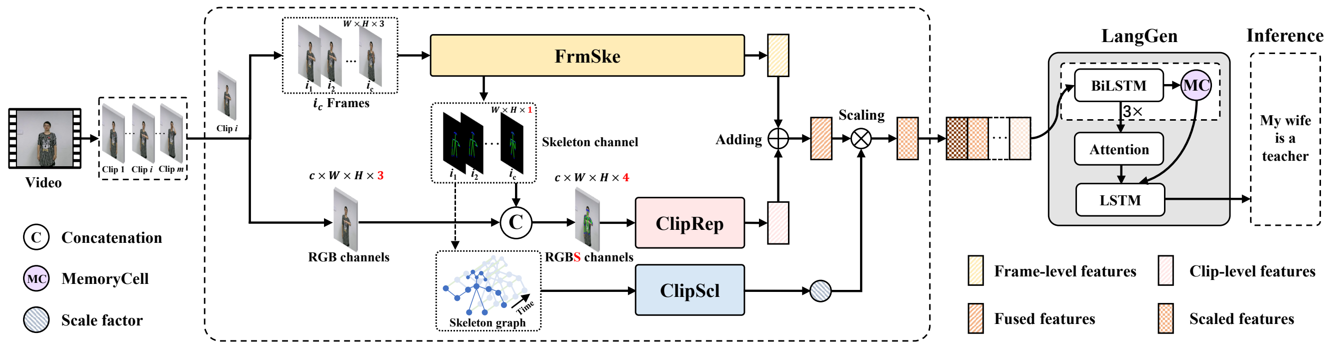


Figure 2: The SANet consists of FrmSke, ClipRep, ClipScl and LangGen, which are used for extracting skeletons and frame-level features, enhancing clip representation, scaling features and generating sentences.

while Camgoz et al. [6] extracted skeleton from video with exiting tool in an offline stage. Then, they parallelly input RGB videos with skeletons to neural network for feature extraction. The external device or offline preprocessing of skeleton extraction hindered the end-to-end SLT from videos. Besides, the existing approaches tended to fuse videos and skeletons in extracted features and give the same importance of each clip, which may limit the performance of feature representation. Differently, we extract skeleton with a self-contained branch to achieve end-to-end SLT. Besides, we fuse videos and skeletons in source data by concatenating skeleton channel and RGB channels, and construct a skeleton-based GCN to weight the importance of each clip.

3 PROPOSED APPROACH

In SLT, when given a sign language video $X = (f_1, f_2, \dots, f_u)$ with u frames, our objective is to learn the conditional probability $p(Y|X)$ of generating a spoken language sentence $Y = (w_1, w_2, \dots, w_v)$ with v words. The sentence with highest probability $p(Y|X)$ is chosen as the translated spoken language.

To realize end-to-end SLT, we propose a Skeleton-Aware neural Network (SANet), which consists of FrmSke, ClipRep, ClipScl and LangGen. As shown in Figure 2, a sign language video is segmented into consecutive equal-length clips with 50% overlap. For a clip, we first use FrmSke to extract skeleton from each frame and frame-level clip features. Then, we concatenate the skeleton channel and RGB channels of each frame in the clip, and adopt ClipRep to extract clip-level features. The frame-level features and clip-level features are added to get the fused features of a clip. Meanwhile, we utilize the skeletons in a clip to construct a spatial-temporal graph and adopt ClipScl to calculate the scale factor, which will be multiplied with the fused features to get the scaled feature vector of a clip. Finally, the scaled features of all clips are sent to LangGen for generating the spoken language.

3.1 Frame-Level Skeleton Extraction

A frame can capture the specific gesture in sign language, thus containing the spatial structure of human pose and detailed information in face, hands, fingers, etc. Therefore, we split the clip into frames, and propose FrmSke module to extract skeleton and frame-level features. As shown in Figure 3, we select a compressed variant

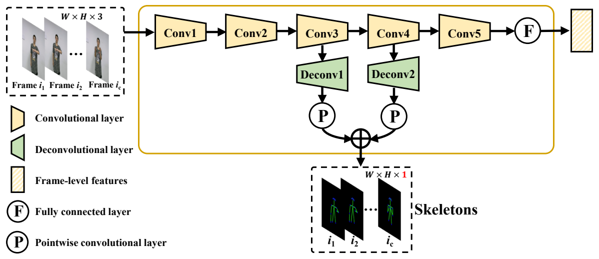


Figure 3: FrmSke extracts skeleton from each image and frame-level features of each clip.

of VGG model [33] as the backbone network for FrmSke. In the compressed VGG model, the number of channels in convolutional layers is reduced to one fourth of the original one, to reduce the memory requirement and make the model work on our platform.

Skeleton extraction: As shown in Figrue 3, to extract the skeleton map from a frame, two parallel deconvolutional networks are used to upsample high-to-low resolution representations [34] after the Conv3 and Conv4 layers. Specifically, Deconv1 layer adopts one 3×3 deconvolution with the stride 2 for 2× upsampling, while Deconv2 layer adopts two consecutive 3×3 deconvolutions with the stride 2 for 4× upsampling. Then, the pointwise convolutional layer and element-sum operation are added after deconvolutional layers to generate K heatmaps, where each heatmap M_k^H, $k \in [1, K]$ contains one keypoint (with the highest heatvalue) of the skeleton. After that, we generate the skeleton (i.e., a 2D matrix) M^S by adding the corresponding elements in K heatmaps. Here, K is set to 14 and means the number of keypoints from nose, neck, both eyes, both ears, both shoulders, both elbows, both wrists and both hips.

Frame-level clip representation: As shown in Figure 3, the convolutions Conv1 to Conv5 are first used to extract feature maps from each frame of a clip. Then, the feature maps are concatenated, flatten and sent to a fully connected layer to get a feature vector F_m with $N_m = 4096$ elements of the clip.

3.2 Channel Extended Clip Representation

A video clip with several consecutive frames can capture the short action (i.e., continuous/dynamic gestures) during sign languages.

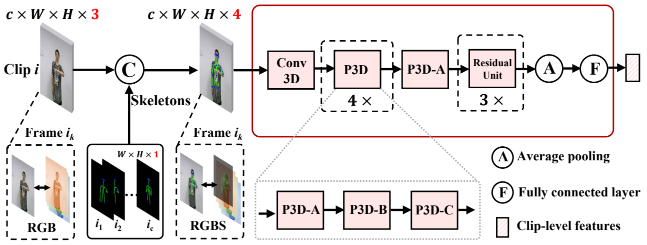


Figure 4: ClipRep extracts clip-level features, where each frame of the clip is extended to four channels by concatenating skeleton channel and RGB channels.

Therefore, we propose ClipRep module to track the dynamic changes of human pose and extract the clip representation. As shown in Figure 4, we first extend the channels of each frame by concatenating the extracted skeleton channel and original RGB channels, and then adopt Pseudo 3D Residual Networks (P3D) [30] to extract clip-level features.

Channel extension with skeleton: We use the skeleton map (i.e., a 2D matrix) as the fourth channel, and concatenate it with original RGB channels of each frame, to get the RGBS frame with four channels, as shown in Figure 4. After that, the clip with RGBS frames will be used for clip-level feature extraction.

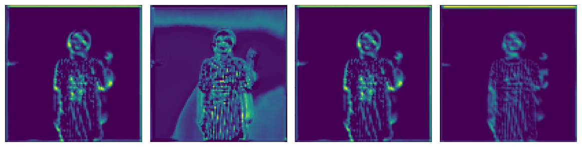
Enhanced clip representation: Based on the RGBS frame sequence of a clip, we introduce the P3D block [30] to extract the features of the clip, where the P3D is first adopted for SLT. For the P3D block, it is consisted of one (2D) spatial filer (1× 3 × 3), one (1D) temporal filter (3× 1× 1), and two pointwise filters (1× 1 × 1). Combining the filters in different ways can get different modules (i.e., P3D-A, P3D-B, P3D-C) for P3D. In ClipRep (shown in Figure 4), after 3D-convolution and P3D blocks, the residual units, average pooling and fully connected layer will be used to get the feature vector F_c with $N_c = 4096$ elements for the clip.

To verify whether the added skeleton channel can enhance feature representation, we visualize the intermediate feature map (i.e., after the first P3D block) in ClipRep without or with using skeleton channel in Figure 5(a) and Figure 5(b), respectively. The areas with brighter colors in Figure 5(b) indicate that the added skeleton channel can highlight the features related to sign language, e.g., gesture changes and human pose, thus enhancing clip representation.

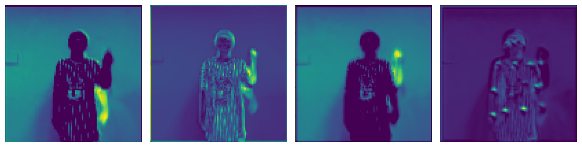
3.3 Skeleton-Aware Clip Scaling

In a short-time clip, the human action can correspond to a meaningful sign, a less important transition action, an unmeaningful end state, etc. Thus the importance of each clip for SLT can be different. To track the human action in a clip and weight the importance of each clip, we propose ClipScl module, which first constructs a skeleton-based graph and applies a Graph Convolutional Network (GCN) [41] to generate a scale factor, and then scales the feature vector of each clip with the scale factor.

Skeleton-based GCN: To track the dynamic changes of human action in a short clip, we construct the skeleton-based graph, which can describe the moving trajectories of keypoints in the skeleton [31, 41]. Specifically, for a clip with c frames, we first construct a skeleton-based graph $G = (V, E)$ with the node set V and



(a) Intermediate feature maps 'NOT' using skeleton



(b) Intermediate feature maps using skeleton

Figure 5: Intermediate feature maps after the first P3D block without or with using skeleton channel. In each case, we show 4 examples selected from 16 frames in a clip.

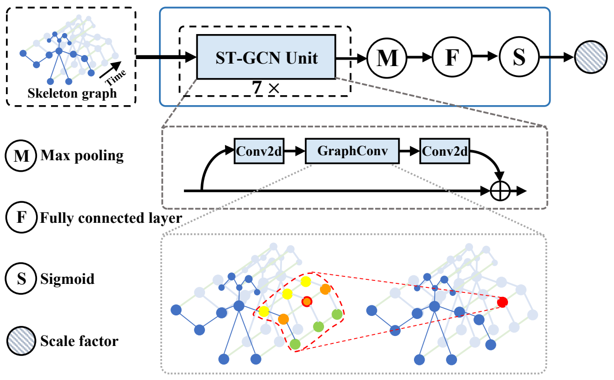


Figure 6: ClipScl constructs skeleton-based graph and uses GCN to calculate the scale factor, which is used for weighting the importance of each clip.

edge set E. Suppose the keypoints of the ith skeleton in a clip are $V_i = (v_{i_1}, v_{i_2}, \dots, v_{i_K}), i \in [1, c]$. Here, $v_{i_j}, j \in [1, K]$ means the jth keypoint/node in the ith skeleton, while $K = 14$ means the number of keypoints in a skeleton. Then we can get the node set $V = \{v_{i_j}, i \in [1, c], j \in [1, K]\}$. In regard to the edge set, it includes the intra-skeleton edge set $E_a = \{v_{i_p}v_{i_q} | (p, q) \in S\}$ where S means the set of naturally connected body joints in a skeleton and the inter-skeleton edge set $E_e = \{v_{i_p}v_{j_p} | i, j \in [1, c], |i - j| = 1\}$ (i.e., edge between the corresponding nodes of two adjacent skeletons), as shown in Figure 6. For each node in the constructed skeleton-based graph, its coordinate vector (x, y) in the frame is used as its initial feature vector v^f.

With the skeleton-based graph,we then adopt Graph Convolutional Network (GCN) to calculate the scale factor (i.e., importance weight) of a clip. Specifically, we design ClipScl, which consists of 7 layers of spatial-temporal graph convolution (ST-GCN) units [41], while decreasing the channel number of ST-GCN by a factor of 0.25 to reduce the memory requirement of the model, as shown in Figure 6. Then, we use the max pooling and a fully connected layer to get the feature vector, which will be passed to a sigmoid function

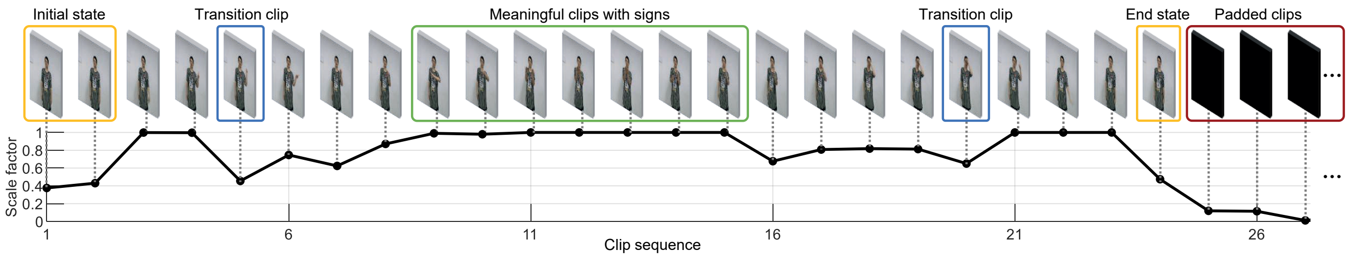


Figure 7: Visualization of the scale factor for each clip. Meaningful clips with signs have larger scale factors, while unmeaningful clips have lower scale factors.

to get the scale factor s_f, i.e., a value belonging to [0, 1].

$$s_f = Sigmoid(ST - GCN^+(V^f, E)) \tag{1}$$

Here, V^f is feature vector set of node set V, $ST - GCN^+(\cdot)$ denotes the combination of 7 layers ST-GCN units, max pooling and a fully connected layer, $Sigmoid(\cdot)$ denotes the sigmoid function.

Fused feature scaling: For each clip, we get the frame-level feature vector F_m, clip-level feature vector F_c and the scale factor s_f. First, we fuse F_m and F_c with element addition $\oplus$ to get the fused feature vector. Then, we scale the fused feature vector with multiplication operation $\otimes$ to get the scaled feature vector F_f, as shown below.

$$F_f = (F_m \oplus F_c) \otimes s_f \tag{2}$$

In Figure 7, we show the calculated scale factor s_f for each clip, where clips with meaningful signs (i.e., clips in the green rectangle) have larger s_f. It means that the designed ClipScl module can efficiently track the dynamic changes of human pose with skeletons and distinguish the importance of different clips, i.e., ClipScl can highlight meaningful clips while weakening unmeaningful clips.

3.4 Spoken Language Generation

After getting the scaled feature vector of each clip, we propose LangGen, which adopts the encoder-decoder framework [35] and attention mechanism [1] to generate the spoken language, as shown in Figure 2.

BiLSTM Encoder with MemoryCell: We use three-layered BiLSTMs and propose a novel MemoryCell to connect adjacent BiLSTM layers for encoding. Specifically, given a sequence of scaled clips' feature vectors $z_{1:n}$, we first get the hidden states $H^l = (h_1^l, h_2^l, \ldots, h_n^l)$ after the lth BiLSTM layer. Then, we design MemoryCell to change the dimensions of hidden states and provide the appropriate input for the following layer, as shown below.

$$h_0^{l+1} = tanh(W \cdot h_n^l + b) \tag{3}$$

where W and b are weight and bias of the fully connected layer, h_n^l is the final output hidden state of the lth layer, h_0^{l+1} is input as the initial hidden state of the $(l+1)$th layer.

LSTM Decoder: We use one LSTM layer as the decoder to decode the word step by step. Specifically, the decoder utilizes LSTM cells, a fully-connected layer and a softmax layer to output the prediction probability $p_{t,j}$, i.e., the probability that the predicted word $\hat{y}_t$ at the tth time step is the jth word in vocabulary. At the beginning of decoding, $\hat{y}_0$ is initialized with the start symbol "[SOS]". At the tth time step, the decoder predicts the word $\hat{y}_t$. The decoder stops decoding until the occurrence of the symbol "[EOS]".

3.5 Joint Loss Optimization

To optimize skeleton extraction and sign language translation jointly, we design a joint loss L, which consists of skeleton extraction loss L_{ske} and SLT loss $L_{slt}(y, \hat{y})$.

The skeleton extraction loss L_{ske} is calculated as follows, where $M^H \in \mathbb{R}^3$, $M^G \in \mathbb{R}^3$ denote the predicted heatmap and the ground-truth heatmap, respectively. Here, K, h, w are the number of keypoints, the height and width of a heatmap. For each ground-truth heatmap, it contains a heating area, which is generated by applying a 2D Gaussian function with 1-pixel standard deviation [34] on the keypoint estimated by OpenPose [8].

$$L_{ske} = \frac{1}{K} \sum_{k}^{K} \sum_{i}^{h} \sum_{j}^{w} (M_{k,i,j}^H - M_{k,i,j}^G)^2 \tag{4}$$

The SLT loss $L_{slt}(y, \hat{y})$ is the cross entropy loss function, $\hat{y}$ is the predicted word sequence and y is the ground-truth word sequence (i.e., labels). The calculation of $L_{slt}(y, \hat{y})$ is shown below, where T means the max number of time steps in decoder and $\mathcal{V}$ means the number of words in vocabulary. $y_{t,j}$ is an indicator, when the ground-truth word at the tth time step is the jth word in vocabulary, $y_{t,j} = 1$. Otherwise, $y_{t,j} = 0$. $p_{t,j}$ means the probability that the predicted word $\hat{y}_t$ at the tth time step is the jth word in vocabulary.

$$L_{slt}(y, \hat{y}) = -\sum_{t=1}^{T} \sum_{j}^{\mathcal{V}} y_{t,j} log(p_{t,j}) \tag{5}$$

Based on L_{ske} and $L_{slt}(y, \hat{y})$, we can calculate the joint loss L as follows, where α is a hyper-parameter and used to balance the ratio of L_{ske} and L_{slt}. We set α to 1 at the beginning of training and change it to 0.5 in the middle of training.

$$L = \alpha L_{ske} + L_{slt}(y, \hat{y}) \tag{6}$$

4 EXPERIMENT

4.1 Datasets

There are two public SLT datasets that are often used, one is the CSL dataset [18] which contains 25K labeled videos with 100 chinese sentences filmed by 50 signers, and the other one is a German sign language dataset: the RWTH-PHOENIX-Weather 2014**T** [5] which contains 8257 weather forecast samples from 9 signers. The

Table 1: Ablation study on CSL dataset. Note: without: 'w/o', skeleton: 'ske', channel: 'chl', graph: 'gph', frame: 'frm', MemoryCell: 'MC', feature: 'fea'. 'all-ske' refers to all skeleton-related components including skeleton extraction branch, skeleton channel and skeleton-based GCN.

Model	Time(s)	ROUGE	BLEU- 1	BLEU-2	BLEU-3	BLEU-4
w/o all-ske	0.467	0.951	0.953	0.939	0.927	0.916
w/o ske-chl	0.489	0.956	0.958	0.946	0.935	0.924
w/o ske-gph	0.472	0.966	0.967	0.957	0.947	0.939
w/o frm-fea	0.480	0.952	0.954	0.941	0.928	0.916
w/o clip-fea	0.242	0.928	0.921	0.898	0.882	0.879
w/o MC	0.489	0.960	0.962	0.950	0.939	0.929
SANet	0.499	**0.996**	**0.995**	**0.994**	**0.992**	**0.990**

Table 2: Ablation study on PHOENIX14T dataset. Note: without: 'w/o', skeleton: 'ske', channel: 'chl', graph: 'gph', frame: 'frm', MemoryCell: 'MC', feature: 'fea'. 'all-ske' refers to all skeleton-related components including skeleton extraction branch, skeleton channel and skeleton-based GCN.

Model	Time(s)	ROUGE	BLEU- 1	BLEU-2	BLEU-3	BLEU-4
w/o all-ske	0.358	0.513	0.542	0.391	0.294	0.225
w/o ske-chl	0.370	0.520	0.549	0.403	0.304	0.233
w/o ske-gph	0.371	0.515	0.545	0.394	0.295	0.226
w/o frm-fea	0.376	0.518	0.547	0.396	0.299	0.230
w/o clip-fea	0.214	0.516	0.541	0.390	0.291	0.220
w/o MC	0.369	0.538	0.563	0.418	0.316	0.243
SANet	0.383	**0.548**	**0.573**	**0.424**	**0.322**	**0.248**

PHOENIX14**T** corpus has two-stage annotations: sign gloss annotations with a vocabulary of 1066 different signs for continuous SLR and German translation annotations with a vocabulary of 2877 different words for SLT. For CSL dataset, we split it into 17K, 2K and 6K for training, validation and testing respectively. The split for CSL dataset was widely adopted in existing work [14, 15, 18]. For PHOENIX14**T**, it has been officially split into 7096, 519 and 642 samples for training, validation, and testing respectively.

4.2 Experimental Setting

In this subsection, we will describe the detailed setting in SANet, including data preprocessing, module parameters, model training, and model implementation. For the data preprocessing, when given a sign language video, each frame of the video is reshaped as 200×200 pixels and Gaussian noises are added for data augmentation. The clip is segmented by using a sliding window where the window size c is set to 16 frames and the stride size s is set to 8 frames. Considering that sign language videos are variable-length, the max/default length of a sign language video is set to 300 frames and 200 frames for CSL dataset and PHOENIX14**T** dataset, respectively. For the module LangGen, the dimensions of BiLSTM and LSTM layers are both 1024. Dropout with rate 0.5 is used after embedding in Decoder/LSTM layer. For model training, the batch size is set to 64. Adam optimizer [19] is used to optimize model parameters with an initial learning rate of 0.001. The learning rate is decreased by a factor of 0.5 every S steps and S is set to the number of training samples. For the model implementation, SANet is implemented with PyTorch1.6 and trained for 100 epochs on 4 NVIDIA Tesla V100 GPUs.

In regard to performance metrics, we adopt ROUGE-L F1-Score [23] and BLEU-1,2,3,4 [26], which are often used to measure the quality of translation in machine translation and also used in the existing SLT work [5–7]. For a fair comparison, we use the evaluation codes of ROUGE-L and BLEU-1,2,3,4 provided by the RWTH-PHOENIX-Weather 2014**T** dataset [5].

4.3 Model Performance

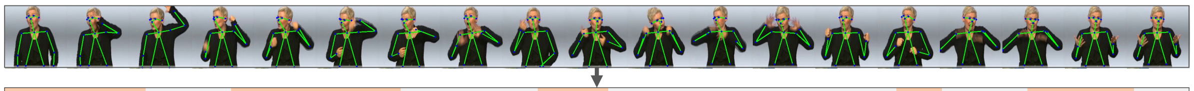
To verify the effectiveness of the proposed Skeleton-Aware neural Network (SANet), we perform ablation study, time analysis and qualitative analysis for SANet.

Ablation Study: To estimate the contributions of our designed components for SLT, we perform the ablation study on two datasets. Specifically, the components include skeleton-related components (i.e., all skeletons, skeleton channel, skeleton-based GCN), feature-related components (i.e., frame-level features, clip-level features) and encoding related component (i.e., MemoryCell). Here, 'all skeletons' means the combination of components related to skeletons, including the branch generating skeletons and the parts using skeletons (i.e., skeleton channel and skeleton-based GCN). In the experiment, we remove only one type of component at a time and list the corresponding performance in Table 1 and Table 2.

According to Table 1 and Table 2, each designed component has made positive contribution to higher performance. For the skeleton-related components, when removing all skeletons, skeleton channel, or skeleton-based GCN, the performance on the metric ROUGE score drops by 4.5%, 4.0%, 3.0% on CSL dataset and 3.5%, 2.8%, 3.3% on PHOENIX14**T** dataset respectively. It indicates that our skeleton-related designs are very helpful in improving performance. That is to say, the proposed self-contained branch can efficiently extract the skeleton, while the designed skeleton channel and skeleton-based GCN can efficiently enhance the feature representation of each clip.

For the feature-related components, they are the important source for encoding and play an important role in SLT. When frame-level features and clip-level features are removed respectively, the performance on the metric ROUGE score drops by 4.4%, 6.8% on CSL dataset and 3.0%, 3.2% on PHOENIX14**T** dataset respectively. It indicates that frame-level features and clip-level features also have a great impact on SLT performance. This is because the frame-level or clip-level features contain rich information extracted from frames or clips. Thus extracting features from frames/clips is a common approach for feature representation in the existing work. However, instead of only extracting features from frames or clips individually, this paper also contributes a design by fusing the frame-level features and clip-level features for a clip. Besides, our work makes a further step on the existing SLT work by introducing skeletons to enhance the feature representation of clips, thus further improve the SLT performance.

In regard to the encoding-related component (i.e., MemoryCell), when MemoryCell is removed, the performance on the metric ROUGE score drops by 3.6% on CSL dataset and 1.0% on PHOENIX14**T**



																					ROUGE
w/o all-ske	am	atlantik	zieht	ein	neues	hoch	über	und	bringt	uns	zum	und	morgenstunden	heftige	schneefälle	auch	teil	auch	mit	regen	55.0%
w/o ske-chl	am	atlantik	zieht	ein	kräftiges	hoch	zur	und	lenkt	uns	zum	morgen	morgenstunden	heftige	schauer	im	teil	auch	kräftigen	regen	50.0%
w/o ske-gph	am	westen	zieht	ein	kräftiges	tief	das	und	lenkt	uns	zum	den	morgenstunden	heftige	schneefälle	heute	teil	auch	mit	regen	65.0%
w/o img-fea	am	westen	zieht	ein	neues	hoch	das	und	lenkt	uns	zum	den	morgenstunden	sonntag	schneefälle	zum	teil	auch	gefrierenden	regen	60.0%
w/o clip-fea	am	westen	zieht	ein	neues	hoch	das	und	lenkt	uns	zum	den	morgenstunden	sonntag	schneefälle	zum	teil	auch	gefrierenden	regen	60.0%
w/o MC	am	atlantik	zieht	ein	tief	tief	heran	und	bringt	uns	zum	und	morgenstunden	heftige	schauer	im	teil	auch	gefrierenden	regen	65.0%
SANet	am	atlantik	zieht	ein	kräftiges	tief	heran	und	bringt	uns	zum	den	morgenstunden	heftige	schneefälle	bringt	teil	auch	gefrierenden	regen	80.0%
Label	vom	nordmeer	zieht	ein	kräftiges	tief	heran	und	bringt	uns	ab	den	morgenstunden	heftige	schneefälle	zum	teil	auch	gefrierenden	regen	

Figure 8: A qualitative analysis of different components used for SLT on the example from PHOENIX14T test set. (Note: the word order in sign language and that in spoken language may be not temporally consistent.)

dataset respectively. It indicates that the designed MemoryCell can efficiently change the dimensions of hidden states between BiLSTM layers and provide the appropriate input for LSTM Decoder, thus contributes to a certain performance improvement for SLT. Therefore, our proposed skeleton-related designs bring a good contribution for SLT, while the overall framework of SANet and other designed components also benefit SLT.

Inference Time Analysis: To evaluate the time efficiency of designed components, in Table 1 and Table 2, we show the inference time of SLT by removing one type of designed component. Here, inference time means the duration of translating the sign language in a video to the spoken language. In the experiment, we evaluate the inference time of a sign language video with 300 frames on CSL dataset and a sign language video with 200 frames on PHOENIX14**T** dataset on a single GPU, and then average the time of 100 runs as the reported inference time. When keeping all components of SANet, the average inference time is 0.499 seconds for CSL dataset and 0.383 seconds for PHOENIX14**T** dataset. When removing any designed component, the interference time decreases. According to Table 1 and Table 2, the clip-level feature extraction introduces much time cost, since 3D convolutions have expensive computational cost. While for the skeleton-related components and MemoryCell, they only have a little effect on the overall inference time.

Qualitative Analysis: In this experiment, we show an example of SLT on a sign language video sample from PHOENIX14**T** test set, to provide a qualitative analysis. As shown in Figure 8, the last row means the ground truth of SLT, the second row from the bottom means the SLT result of the proposed SANet, while other rows mean the SLT results by removing the designed components of SANet. If the ith word in the SLT result is different from the ith ground-truth word, it is an error and marked with orange color. According to Figure 8, the proposed SANet achieves the best performance, while removing any designed component will lead to more errors. It demonstrates that the proposed framework and all the designed components contribute to a higher performance for SLT.

4.4 Comparisons

Evalutation on CSL dataset: We compare SANet with existing approaches on two settings. (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 sentences of training set and test set are the same, but the signers have no overlaps. (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 Table 3, we show the SLT performance on Split I and Split II, and also compare our SANet with the following existing approaches: (1) **S2VT** [39] belongs to a standard two-layers stacked LSTM architecture which is used to translate video to text. (2) **S2VT(3-layer)** extends the S2VT from two-layers LSTM to three-layers LSTM. (3) **HLSTM** [15] is a hierarchical LSTM based encoder-decoder model for SLT. (4) **HLSTM-attn** [15] adds the attention mechanisms over the previous HLSTM. (5) **HRF-Fusion** [14] is a hierarchical adaptive recurrent network which mines variable-length key clips and applies attention mechanisms, by using both RGB videos and skeleton data from Kinect.

As shown in Table 3, on Split I, whatever the performance metric is, the proposed SANet achieves the best performance. Specifically, the proposed SANet achieves 99.6%, 99.4%, 99.3%, 99.2% and 99.0% on ROUGE, BLEU-1, BLEU-2, BLEU-3 and BLEU-4 respectively, which outperform the existing approaches. When compared with S2VT, the SANet can increase the ROUGE, BLEU-1, BLEU-2, BLEU-3 and BLEU-4 by 9.2%, 9.2%, 10.7%, 11.3% and 11.6%, respectively. When moving to Split II, the performance drops a lot. Take our SANet as an example, the ROUGE score on Split II drops by 31.5%. This is because translating the unseen sentences, i.e., the words in the sentence exist in the training set while the sentence does not occur in the training set, can be more challenging and it is difficult for SLT. However, our SANet still outperforms the state-of-the-art approaches and achieves 68.1%, 69.7%, 41.1%, 26.8% and 18.1% on ROUGE, BLEU-1, BLEU-2, BLEU-3 and BLEU-4, respectively. When compared with the existing approach HRF-Fusion [14], our SANet can increase ROUGE by 23.2%.

Evaluation on PHOENIX14T dataset: We compare SANet with the following existing approaches: (1) **TSPNet** [22] introduces multiple segments of sign language video in different granularities and uses Transformer decoder for SLT. (2) **H+M+P** [6] uses a multi-channel transformer architecture, by fusing hand-area videos, mouth-area videos and extracted poses from videos. (3) **Sign2Gloss→Goss2Text** [5] adopts a classic encoder-decoder architecture, and it is trained in two stages (i.e., gloss stage and sentence stage) independently. (4) **Sign2Gloss2Text** [5] adopts the

Table 3: Comparisons with other approaches on CSL dataset under signer-independent test and unseen-sentences test.

Model	Split I					Split II				
	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4
S2VT [39]	0.904	0.902	0.886	0.879	0.874	0.461	0.466	0.258	0.135	-
S2VT(3-layer) [39]	0.911	0.911	0.896	0.889	0.884	0.465	0.475	0.265	0.145	-
HLSTM [15]	0.944	0.942	0.932	0.927	0.922	0.481	0.487	0.315	0.195	-
HLSTM-attn [15]	0.951	0.948	0.938	0.933	0.928	0.503	0.508	0.330	0.207	-
HRF-Fusion [14]	0.994	0.993	0.992	0.991	0.990	0.449	0.450	0.238	0.127	-
SANet	**0.996**	**0.994**	**0.993**	**0.992**	**0.990**	**0.681**	**0.697**	**0.411**	**0.268**	**0.181**

Table 4: Comparison with other approaches on RWTH-PHOENIX-Weather 2014T dataset

Model	PHOENIX14**T** DEV					PHOENIX14**T** TEST				
	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	BLEU-1	BLEU-2	BLEU-3	BLEU-4
TSPNet [22]	-	-	-	-	-	0.349	0.361	0.231	0.169	0.134
H+M+P [6]	0.459	-	-	-	0.195	0.436	-	-	-	0.183
Sign2Gloss→Goss2Text [5]	0.438	0.411	0.291	0.221	0.179	0.435	0.415	0.295	0.222	0.178
Sign2Gloss2Text [5]	0.441	0.429	0.303	0.230	0.184	0.438	0.433	0.304	0.228	0.181
(Gloss)Sign2Text [7]	-	0.455	0.326	0.253	0.207	-	0.453	0.323	0.248	0.202
(Gloss)Sign2(Gloss+Text) [7]	-	0.473	0.344	0.271	0.224	-	0.466	0.337	0.262	0.213
DeepHand [25]	-	-	-	-	-	0.381	0.385	0.256	0.186	0.146
SANet	**0.542**	**0.566**	**0.415**	**0.312**	**0.235**	**0.548**	**0.573**	**0.424**	**0.322**	**0.248**

previous encoder-decoder architecture, but it is trained jointly. (5) **(Gloss)Sign2Text** [7] utilizes a transformer based architecture for SLT. (6) **(Gloss)Sign2(Gloss+Text)** [7] also utilizes a transformer based architecture, but jointly learns CLSR and SLT at the same time. (7) **DeepHand** [25] adopts the encoder-decoder architecture and introduces a pretrained hand shape recognizer for SLT.

As shown in Table 4, we provide the performance of each approach on both the validation set (i.e., ‘DEV’) and test set (i.e., ‘TEST’). Whatever on ‘DEV’ set or ‘TEST’ set, the existing approaches often have the lower performance. For example, the best ROUGE and BLEU-4 score achieved by the existing approach is less than 46% and 23%, respectively. This may be because of the high diversity, large size of vocabulary and limited number of training samples in the PHOENIX14**T** dataset. However, our proposed SANet further improves the SLT performance and achieves the best performance, e.g., 54.2% of ROUGE and 23.5% of BLEU-4 score on ‘DEV’ set while 54.8% of ROUGE and 24.8% of BLEU-4 score on ‘TEST’ set, which outperform all the existing approaches.

5 DISCUSSION AND FUTURE WORK

Keypoints of skeleton: In this paper, we extract 14 keypoints to represent the skeleton, while ignoring the fine-grained keypoints in fingers, mouth, etc. The main reason is that it is difficult to accurately extract fine-grained keypoints from the limited-resolution frames in SLT dataset (e.g., the frame in PHOENIX14T dataset is only 210×260 pixels). However, we consider that accurately extracting more fine-grained keypoints can advance SLT performance. In the future, we will make further research on skeleton extraction.

More comparisons from different perspectives: In the experiment, we compare the proposed SANet with the existing approaches in terms of ROUGE and BLEU-1,2,3,4, which are used to evaluate the translation quality. For a fair comparison, the reported results on these metrics are provided by the original paper. Without complete original/released codes, we do not make further comparisons (e.g., inference time) with the approaches. However, in future work, we will try to explore more comprehensive comparisons from different perspectives with the existing approaches.

Particularity of CSL dataset: The CSL dataset is a special dataset, where the word order of sign language is temporally consistent with that of spoken language, thus it can be used for SLR task [18, 45] as well as SLT task [14, 15]. In this paper, CSL dataset is solved from the aspect of SLT and only SLT methods are considered for comparisons.

6 CONCLUSION

In this paper, we propose a Skeleton-Aware neural Network (SANet) for end-to-end SLT. Specifically, we first use a self-contained branch to extract the skeleton from each frame, and then enhance the feature representation of a clip by adding the skeleton channel and scaling (i.e., weighting the importance) the feature vector with a designed skeleton-based GCN. Besides, we design a joint optimization strategy for training. The experimental results on two large scale datasets demonstrate the effectiveness of SANet.

7 ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China under Grant No. 2018AAA0102302; National Natural Science Foundation of China under Grant Nos. 61802169, 61872174, 61832008, 61906085, 61902175; JiangSu Natural Science Foundation under Grant Nos. BK20180325, BK20190293. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Kshitij Bantupalli and Ying Xie. 2018. American sign language recognition using deep learning and computer vision. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 4896–4899.

[3] 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.

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

[5] 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

[6] 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).

[7] 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.

[8] 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.

[9] Xiujuan Chai, Guang Li, Yushun Lin, Zhihao Xu, Yili Tang, Xilin Chen, and Ming Zhou. 2013. Sign language recognition and translation with kinect. In *IEEE Conf. on AFGR*, Vol. 655. 4.

[10] Runpeng Cui, Hu Liu, and Changshui Zhang. 2019. A deep neural framework for continuous sign language recognition by iterative training. *IEEE Transactions on Multimedia* 21, 7 (2019), 1880–1891.

[11] Amanda Duarte, Shruti Palaskar, Lucas Ventura, Deepti Ghadiyaram, Kenneth DeHaan, Florian Metze, Jordi Torres, and Xavier Giro-i Nieto. 2021. How2Sign: A Large-scale Multimodal Dataset for Continuous American Sign Language. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] 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

[13] Dan Guo, Shuo Wang, Qi Tian, and Meng Wang. 2019. Dense Temporal Convolution Network for Sign Language Translation.. In *IJCAI*. 744–750.

[14] 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.

[15] Dan Guo, Wengang Zhou, Houqiang Li, and Meng Wang. 2018. Hierarchical lstm for sign language translation. In *AAAI*, Vol. 32.

[16] Dan Guo, Wengang Zhou, Meng Wang, and Houqiang Li. 2016. Sign language recognition based on adaptive hmms with data augmentation. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2876–2880.

[17] 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.

[18] Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, and Weiping Li. 2018. Video-based sign language recognition without temporal segmentation. In *AAAI*.

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] 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).

[21] 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.

[22] 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 *Advances in Neural Information Processing Systems*, Vol. 33.

[23] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[24] Tao Liu, Wengang Zhou, and Houqiang Li. 2016. Sign language recognition with long short-term memory. In *ICIP*. IEEE, 2871–2875.

[25] Alptekin Orbay and Lale Akarun. 2020. Neural sign language translation by learning tokenization. *arXiv preprint arXiv:2002.00479* (2020).

[26] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*. 311–318.

[27] Lionel Pigou, Mieke Van Herreweghe, and Joni Dambre. 2017. Gesture and sign language recognition with temporal residual networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 3086–3093.

[28] Junfu Pu, Wengang Zhou, Hezhen Hu, and Houqiang Li. 2020. Boosting Continuous Sign Language Recognition via Cross Modality Augmentation. In *Proceedings of the 28th ACM International Conference on Multimedia*. 1497–1505.

[29] Junfu Pu, Wengang Zhou, and Houqiang Li. 2019. Iterative alignment network for continuous sign language recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4165–4174.

[30] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *CVPR*. 5533–5541.

[31] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. 2019. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7912–7921.

[32] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199* (2014).

[33] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[34] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep high-resolution representation learning for human pose estimation. In *CVPR*. 5693–5703.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. 3104–3112.

[36] Ao Tang, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li. 2015. A real-time hand posture recognition system using deep neural networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 2 (2015), 1–23.

[37] Dominique Uebersax, Juergen Gall, Michael Van den Bergh, and Luc Van Gool. 2011. Real-time sign language letter and word recognition from depth data. In *2011 IEEE international conference on computer vision workshops (ICCV Workshops)*. IEEE, 383–390.

[38] 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.

[39] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *ICCV*. 4534–4542.

[40] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*. Springer, 20–36.

[41] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*.

[42] Siyuan Yang, Jun Liu, Shijian Lu, Meng Hwa Er, and Alex C Kot. 2020. Collaborative learning of gesture recognition and 3D hand pose estimation with multi-order feature analysis. In *European Conference on Computer Vision*. Springer, 769–786.

[43] Zhaoyang Yang, Zhenmei Shi, Xiaoyong Shen, and Yu-Wing Tai. 2019. SF-Net: Structured Feature Network for Continuous Sign Language Recognition. *arXiv preprint arXiv:1908.01341* (2019).

[44] Jihai Zhang, Wengang Zhou, and Houqiang Li. 2014. A threshold-based hmm-dtw approach for continuous sign language recognition. In *ICIMCS*. 237–240.

[45] Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. 2020. Spatial-Temporal Multi-Cue Network for Continuous Sign Language Recognition.. In *AAAI*. 13009–13016.

[46] Hao Zhou, Wengang Zhou, Yun Zhou, and Houqiang Li. 2021. Spatial-temporal multi-cue network for sign language recognition and translation. *IEEE Transactions on Multimedia* (2021).