Spatial-Temporal Graph Convolution-Transformer に基づく手話認識 *

高 山 夏 樹 ** Gibran BENITEZ-GARCIA** 高 橋 裕 樹 **,***

Sign Language Recognition Based on Spatial-Temporal Graph Convolution-Transformer

Natsuki TAKAYAMA, Gibran BENITEZ-GARCIA and Hiroki TAKAHASHI

This paper reports on sign language recognition based on human body part tracking. Tracking-based sign language recognition has practical advantages, such as robustness against variations in clothes and scene backgrounds. However, there is still room for improving feature extraction in tracking-based sign language recognition. In this paper, a tracking-based continuous sign language word recognition method called Spatial-Temporal Graph Convolution-Transformer is presented. Spatial-temporal graph convolution is employed to improve framewise feature extraction using tracking points, while Transformer enables the model to recognize word sequences of arbitrary lengths. Besides the model design, the training strategy also has an impact on the recognition performance. Multi-task learning, which combines connectionist temporal classification and cross-entropy losses, is employed to train the proposed method in this study. This training strategy improved the recognition performance by a significant margin. The proposed method was evaluated statistically using a sign language video dataset consisting of 275 types of isolated words and 120 types of sentences. The evaluation results show that STGC-Transformer with multi-task learning achieved 12.14% and 2.07% word error rates for isolated words and sentences, respectively.

Key words: deep neural networks, multi-task learning, sign language recognition, spatial temporal graph convolution, transformer

1. Introduction

Sign language translation, which refers to the translation of native signers' signs into text, contributes to improving communications between native signers and speakers. Sign language is commonly represented by several visual cues, such as hand motion and shapes, and non-manual signals, including posture, facial expression, gaze, and mouth motions. Owing to this characteristic, vision-based sign language recognition, in which signs are recognized from videos, is an important technique for improving sign language translation.

Sign language recognition has a history of about 30 years. Research efforts have led to proposals for continuous sign language word recognition [1) 2)] and translation [3) 4)] in recent years. Convolutional neural networks (CNNs) are directly applied to video frames to extract framewise features in these methods.

At the same time, tracking-based sign language recognition has become feasible because of recent improvements in human body part tracking. Tracking-based sign language recognition has practical advantages. For example, the tracking points are robust against scene variation, while lightweight data reduce the learning time and storage capacity required. Moreover, the tracking can also work on edge devices. These characteristics are essential for large-scale sign language recognition and production systems. Therefore, it is expected that the importance of tracking-based sign language recognition will increase in the future.

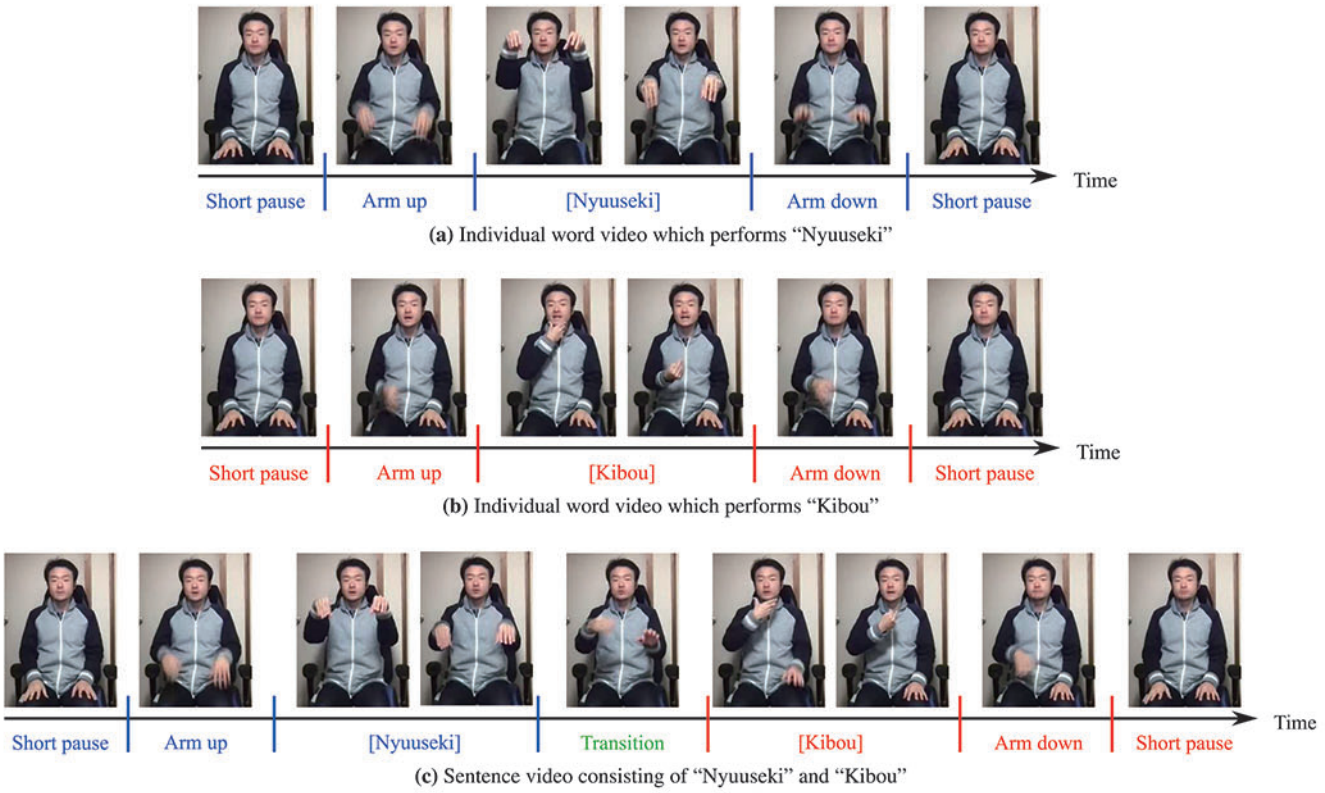
Owing to the above considerations, we propose a tracking-based sign language recognition method in this paper. The tracking points are more abstracted data than raw images. Therefore, feature extraction performance is a crucial issue in the approach. Moreover, continuous sign language recognition must be able to handle variable-length sequences, unlike standard action recognition.

To address these issues, we propose a tracking-based sign language recognition referred to as Spatial-Temporal Graph Convolution (STGC)-Transformer. We employ STGC [5)] to improve framewise feature extraction from the tracking points. Furthermore, we combine STGC with Transformer [6)] to recognize the arbitrary lengths of continuous sign language words.

In addition to the model design, the training strategy is also an essential part of deep neural network (DNN) models. We apply multi-task learning, which combines the connectionist temporal classification (CTC) [7)] and cross-entropy (CE) losses to train the proposed model. The effectiveness of this training strategy is demonstrated through evaluation.

In this paper, we report the recognition performance using a sign language video dataset. The dataset contains 275 types of isolated word videos and 120 types of sequence videos. An example of a sign-language video is shown in **Fig.1**. Figure **1** (a) and (b) are examples of the isolated word videos which perform "入籍 (Nyuuseki)" and "希望 (Kibou)." "Nyuuseki" and "Kibou" mean "registration of marriage" and "hope" in Japanese sign language, respectively. The videos include marginal motions such as "Short pause," "Arm up," and "Arm down." These marginal motions do

* 原稿受付 令和3年5月12日
掲載決定 令和3年7月15日
** 電気通信大学大学院情報理工学研究科
(東京都調布市調布ケ丘 1-5-1)
*** 電気通信大学人工知能先端研究センター (同上)



(a) Individual word video which performs "Nyuuseki"

(b) Individual word video which performs "Kibou"

(c) Sentence video consisting of "Nyuuseki" and "Kibou"

Fig. 1 Examples of sign language videos

not have lexical meanings. Figure **1** (c) is an example of a sentence video consisting of "Nyuuseki" and "Kibou," which means "I'd like to register our marriage." The sequence is represented by expressing the words continuously. The "Transition" motion appears between each word motion. Although the "Transition" motion is dependent on the neighboring word motions, it does not have lexical meaning.

Garbage modeling [8] and attention networks [6,9] are two well-known methods for handling marginal motions. In garbage modeling, marginal motions are recognized explicitly and removed in post-processing. In contrast, the attention network learns to attenuate the effects of marginal motions on recognition. Garbage modeling requires the marginal motions to be manually designed, for which the selection of appropriate parameters is difficult. Therefore, we employ Transformer, which includes an attention network in its block.

Isolated word recognition and continuous word recognition have different characteristics. As shown in Fig.**1**, the motion for a sentence is more complicated than that for an isolated word. On the other hand, additive information, such as variations in sequence length and the label order, are present in sentences. Previous methods have targeted either isolated word recognition or continuous word recognition and lack a unified evaluation. The evaluation of the proposed method for both isolated and continuous word recognition is reported in this paper.

The remainder of this paper is organized as follows. In Section 2, we introduce conventional sign language recognition, tracking-based feature extraction, and multi-task learning methods. In Section 3, we describe STGC and its implementation in this study. In Section 4, we describe the proposed sign language recognition method. In Section 5, we describe our sign-language video dataset and evaluate the proposed method. Finally, we give our conclusions and suggestions for future research in Section 6.

2. Related Work

2.1 Sign Language Recognition

One of the standard sign language recognition is a combination of framewise feature extraction and temporal recognition. The early methods employ handcrafted features such as SIFT and HOG [10)–12)] and combine them with statistical temporal pattern recognition methods, such as the hidden Markov model (HMM). However, these handcrafted features do not give adequate performance, and the HMM requires careful manual design. For these reasons, these elements have been replaced by DNN, and many types of architectures [2)–4),8),13)] have been proposed in recent years.

Coster et al. [14] proposed a Transformer-based method for tracking-based sign language recognition. They applied OpenPose [15] to extract tracking points and combined the tracking points with CNN features to enrich the framewise features. Li et al. [13] proposed temporal graph convolutional networks for performance comparison. Similar to our approach, they attempted to improve feature extraction by using graph convolution and Transformer. However, they focused on isolated word recognition in which the entire sequence was utilized to recognize each word. Therefore, their methods cannot be easily applied for continuous word recognition. STGC-Transformer is designed for continuous word recognition and can be applied to isolated words to recognize single word sequences.

2.2 Tracking-Based Feature Extraction

Feature extraction using tracking points has been studied in the field of action recognition. Tracking-based feature extraction methods can be divided into handcrafted and DNN-based approaches. Handcrafted approaches include normalized relative positions [16], sequential position covariance [17], geometric transformation parameters [18], and directional vectors between neighboring points [19) 20)]. As for DNN-based methods, recurrent neural networks (RNNs) have often been applied to capture temporal dependencies [21) 22)]. The standard CNN has also been used for tracking sequences encoded in two-dimensional images [23) 24)]. Similar to image features, handcrafted features do not give sufficient performance. RNN and CNN cannot easily capture the local relationships between the tracking points correctly.

Graph convolution is a generalization of CNN to a graph structure. Yan et al. [5] proposed STGC, in which graph convolution is applied to a human skeleton. STGC is a hot topic in action recognition and has been extended to body parts-based subgraphs [25] and multi-stream structures with attention networks [26) 27)]. We note that the original STGC and its extensions use the entire sequence to recognize a single action, similar to isolated word recognition [13) 14)].

2.3 Multi-Task Learning

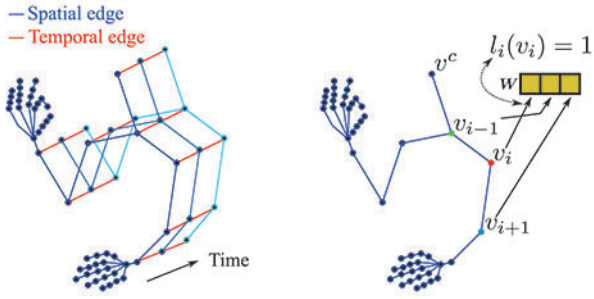
Multi-task learning is a training approach for generalizing models by combining multiple losses associated with each task [28]. Combinations of alignment, framewise decoding, and sequence-to-sequence decoding losses have often been applied in sign language recognition [4) 29)–33)]. In this study, we employ a combination of framewise and sequence-to-sequence decoding losses because using the alignment loss often requires complicated training procedures, such as multi-stage or iterative optimizations [29) 30) 32)].

3. Framewise Feature Extraction Based on STGC

3.1 Graph Construction

A brief introduction to STGC and its implementation in our study is presented in this section. Graph convolution is a generalization of CNN that performs convolution on graph-structured data, and it calculates a weighted sum of features of one node and its neighbors. In STGC, features are extracted by applying spatial and temporal graph convolution to the spatiotemporal graph $G = (\boldsymbol{V}, \boldsymbol{E})$. $\boldsymbol{V} = \{v_{ti} | t = 1, \ldots, T; i = 1, \ldots, N\}$ indicates a set of nodes where T and N are the numbers of video frames and tracking points, respectively. Each node is associated with each tracking point in a frame. $\boldsymbol{E} = \boldsymbol{E}^N \bigcup \boldsymbol{E}^T \bigcup \boldsymbol{E}^I$ indicates a set of edges where $\boldsymbol{E}^N = \{(v_{ti}, v_{tj}) | i \neq j\}$ is a set of spatial edges that connect nodes in the same frame, $\boldsymbol{E}^T = \{(v_{ti}, v_{(t+1)i})\}$ is a set of temporal edges that connect the nodes which refer same tracking points along the time axis, and $\boldsymbol{E}^I = \{(v_{ti}, v_{ti})\}$ is a set of self-connections. $\boldsymbol{E}^N$ and $\boldsymbol{E}^T$ define neighbors of nodes in spatial and temporal graph convolution, respectively. $\boldsymbol{E}^I$ is used in both spatial and temporal graph convolution to refer to self features of nodes.

We used 50 tracking points on the nose, neck, shoulders, arms, and hands as the input to STGC. We employed OpenPose [15] for human body parts tracking. The features from each tracking point include its horizontal and vertical coordinates.



(a) Graph structure (b) Mapping joints to weights

Fig. 2 Spatiotemporal graph for STGC

Let node $v_i; i = 1, \ldots, 50$ be associated with the tracking points obtained using OpenPose. In the proposed method, the spatial edges are connected based on the human skeleton, and the nodes which refer same tracking points in neighboring frames are connected by temporal edges, as shown in **Fig.2(a)**. The blue circles in Fig.**2**(a) indicate the nodes. The blue lines are the spatial edges, where edges belonging to the same frame are indicated with the same saturation. The red lines are the temporal edges. The temporal edges on the hand and the self-connections are omitted in Fig.**2**(a) for better visibility.

3.2 Graph Convolution

The spatial graph convolution for node v_i is written as

$$\boldsymbol{f}_{out}(v_i) = \sum_{v_j \in \boldsymbol{B}^N(v_i)} \frac{1}{Z_{ij}} \boldsymbol{f}_{in}(v_j) \boldsymbol{w}(l_i(v_j)), \tag{1}$$

where $\boldsymbol{f}(v)$ denotes the feature vector of the node, and $\boldsymbol{B}^N(v_i) = \{v_j | d(v_j, v_i) \leq D)\}$ is the neighborhood of node v_i. $d(v_i, v_j)$ indicates the shortest distance between two nodes. D is a control parameter that defines the neighborhood. We used $D = 1$ in this study. $\boldsymbol{w}$ is a function that returns the weight based on the index value. l_i is a mapping function that associates a node with a weight vector as follows:

$$l_i(v_j) = \begin{cases} 1; & d(v^c, v_j) = d(v^c, v_i) \\ 2; & d(v^c, v_j) < d(v^c, v_i) \\ 3; & d(v^c, v_j) > d(v^c, v_i). \end{cases} \tag{2}$$

v^c is the reference point of the mapping function, and $Z_{ij} = \sum_{v_k} \deg(v_k); l_i(v_k) = l_i(v_j)$ is a normalization term that adjusts the effect of the node on the output. $\deg(v)$ denotes the degree of the node.

We employ the tracking point on the nose as the reference point v_c. Figure **2** (b) shows an example of mapping between node v_i and its neighbors.

Similar to the spatial graph convolution, the temporal graph convolution for node v_{ti} is written as

$$\boldsymbol{f}_{out}(v_{ti}) = \sum_{v_{\tau i} \in \boldsymbol{B}^T(v_{ti})} \boldsymbol{f}_{in}(v_{\tau i}) \boldsymbol{w}(l_t(v_{\tau i})), \tag{3}$$

where $\boldsymbol{B}^T(v_{ti}) = \{v_{\tau i} | d(v_{\tau i}, v_{ti}) \leq \lfloor K_t/2 \rfloor\}$ is the neighborhood of node v_{ti} along temporal edges. K_t indicates a temporal

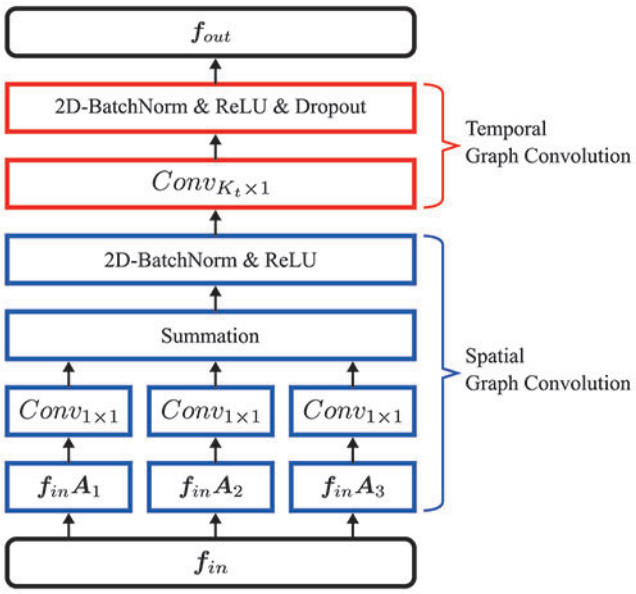


Fig. 3 Process flow of STGC

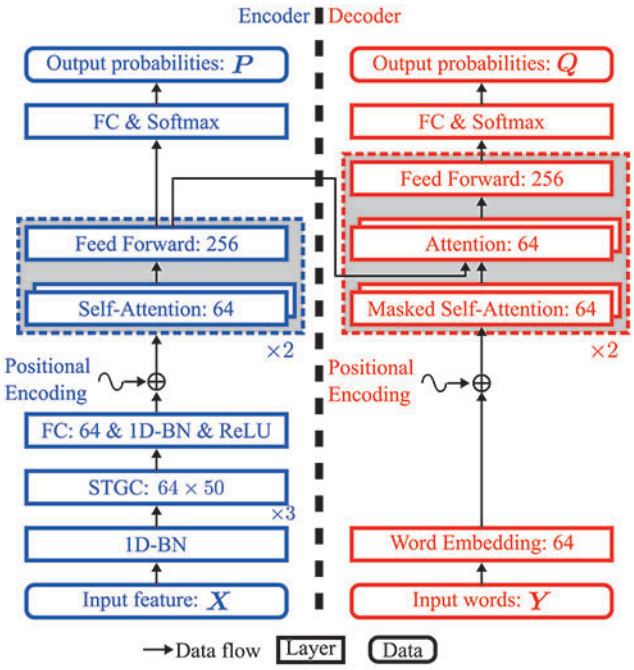


Fig. 4 Overview of the recognition model

range that defines the neighborhood. $l_t(v_{\tau i}) = \tau - t + \lfloor K_t/2 \rfloor$ is a mapping function. The mapping function $l_t(v_{\tau i})$ associates nodes with weight vectors one by one along the time axis for the temporal graph convolution.

3.3 Implementation

The implementation of STGC is described as follows: We employ cascaded architecture, consisting of separated spatial and temporal graph convolution layers similar to the extended methods [26) 27)] rather than a unified spatial-temporal graph convolution layer [5)]. **Figure 3** shows the process flow of STGC. Blue and red colors represent the spatial and temporal graph convolutions, respectively. Let C be the feature dimensions belonging to each node. $\boldsymbol{f} \in \mathcal{R}^{C\times T\times N}$ represents the feature map of the entire sequence. At this time, the spatial graph convolution can be converted to

$$\sum_k^K \boldsymbol{W}_k(\boldsymbol{f}_{in}\boldsymbol{A}_k), \tag{4}$$

where $K = 3$ indicates the kernel size determined by the mapping function l_i, and $\boldsymbol{A}_k = \boldsymbol{\Lambda}_k^{-1/2}(\hat{\boldsymbol{A}}_k \otimes \boldsymbol{M})\boldsymbol{\Lambda}_k^{-1/2}$ is an $N \times N$ matrix. $\hat{\boldsymbol{A}}_k$ is a subset of the adjacency matrix determined by the mapping function l_i. $\boldsymbol{\Lambda}_k$ is an $N \times N$ matrix with $\boldsymbol{\Lambda}_k^{ij} = \sum_j(\hat{\boldsymbol{A}}_k^{ij}) + \alpha$ as the matrix elements. $\alpha = 0.001$ is a constant included to avoid division by zero. $\boldsymbol{M}$ is an $N \times N$ matrix that applies weights to the spatial edges. The elements of $\boldsymbol{M}$ are initialized to 1 and updated through training. $\otimes$ denotes the Hadamard product. $\boldsymbol{W}_k(\cdot)$ is a pointwise convolution that has different parameters depending on k.

The feature vectors are mapped to weight vectors one by one as described in the mapping function $l_t(v_{\tau i})$. Moreover, the feature vectors are well-ordered in the feature map $\boldsymbol{f}$. Therefore, the temporal graph convolution can be implemented by a standard convolution layer with a $K_t \times 1$ kernel. We used $K_t = 9$ in this work. Batch Normalization, ReLU, and Dropout were applied to stabilize the training.

4. Sign Language Recognition Based on STGC-Transformer

4.1 Process Overview

Continuous sign language word recognition can be modeled as a sequence-to-sequence learning problem. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T\}$, $\boldsymbol{x}_t \in \mathcal{R}^{100}$ and $\boldsymbol{Y} = \{y_1, \ldots, y_s, \ldots, y_S\}$, $y_s \in \{<\text{start}>, <\text{end}>, <\text{pad}>, \boldsymbol{W}\}$ be the input feature sequence and word sequence, respectively. $\boldsymbol{x}_t$ indicates a set of tracking coordinates extracted from the tth frame. $\boldsymbol{W}$ is the set of words to be recognized. <start> and <end> represent the start and end of the sentence, respectively. <pad> indicates a padding keyword to ensure that the lengths of the sentences are the same. We note that the marginal motions, "Short pause," "Arm up," "Arm down," and "Transition" are not included in the vocabulary. $\boldsymbol{Y}$ is represented as a sequence of discrete indices in the implementation. STGC-Transformer attempts to learn the optimized projection $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ through training.

The input feature sequences are required to have a fixed length during both training and testing. Similarly, the input word sequences also need to have a fixed length during training. These requirements are due to the limitations of the mini-batch training and the Transformer layers. Let T_{max} and S_{max} be the maximum lengths of the input feature sequences and input word sequences in the dataset, respectively. If the length of a feature sequence is $T < T_{max}$, $\boldsymbol{0} \in \mathcal{R}^{100}$ is inserted after $\boldsymbol{x}_T$. Similarly, if the length of a sequence is $S < S_{max}$, $< pad >$ is inserted after y_S.

Figure 4 shows the overview of the recognition model. The proposed model consists of an encoder and decoder. The encoder transforms the input sequence into the abstracted vector sequence. The decoder utilizes the output from the encoder and input word sequence to predict the output word sequentially. Figure **4** illus-

trates the encoder and decoder with blue and red colors, respectively. The rectangles and rounded rectangles indicate the layers and data, respectively. The number of rectangles drawn for each layer indicates the dimension of the output from the layer. FC and 1D-BN are fully connected layers and temporal batch normalization layers, respectively. The gray rectangles indicate the Transformer encoder and decoder blocks. We employ two heads of attention networks in the Transformer blocks. Two Transformer blocks are cascaded in the encoder and decoder.

The FC layers followed by softmax layers convert the intermediate features into the output probabilities of the words. The vocabulary determines the units of the final FC layers in the encoder and decoder.

4.2 Encoding Process

The encoder transforms the input sequence into the abstracted vector sequence. First, the encoder normalizes the coordinates using temporal batch normalization. Next, three cascaded STGC layers apply graph convolution to the intermediate features. As described in Section 3, the intermediate features outputted by STGC are $C \times T \times N$ feature maps. Generally, the input to Transformer is assumed to be a $T \times C$ feature vector. Simply flattening the feature map will give $64 \times 50 = 3200$-dimensional feature vectors, which are expected to contain redundancy. Therefore, the fully connected layer is applied to the intermediate features after STGC, and the intermediate features are converted into 64-dimensional vectors.

After the framewise feature transformation, multidimensional-temporal signals are added to the features by positional encoding. The multidimensional-temporal signals are defined as

$$\begin{aligned} PE(t,2k) &= \sin \frac{t}{10000^{2k/d_{model}}}, \\ PE(t,2k+1) &= \cos \frac{t}{10000^{2k/d_{model}}}, \end{aligned} \tag{5}$$

where d_{model} denotes the dimensions of the input features. We set $d_{model} = 64$ for the encoder and decoder in this work.

The output of the Transformer encoder has two branches. One branch is connected to the Transformer decoder. The other branch is directed to the FC layer. The final FC and softmax layers of the encoder convert the intermediate features into the frame-wise word probabilities.

The calculation of the framewise decoding loss based on CTC is as follows: Let $\boldsymbol{P} = \{\boldsymbol{p}_t\}; \boldsymbol{p}_t \in [0.0, 1.0]^{\times|\boldsymbol{W}|+1}$ be the output probabilities of the encoder, where $|\boldsymbol{W}|$ is the number of vocabularies. The additional dimension of $\boldsymbol{p}_t$ corresponds to the "blank" keyword of CTC. CTC regards redundant representations, including "blank" and repeated words, as identical sequences and seeks the most probable sequences to the target sequence. Let $\boldsymbol{\pi}$ be a path that matches a redundant representation of $\boldsymbol{Y}$. The CTC loss is then calculated as

$$L_{CTC} = -\frac{1}{N} \log p(\boldsymbol{Y}|\boldsymbol{X}), \tag{6}$$

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{\boldsymbol{\pi} \in \boldsymbol{\Omega}} \sum_{t} \log p_t^{\pi t}, \tag{7}$$

where $\boldsymbol{\Omega}$ is a set of paths that correspond to $\boldsymbol{Y}$ and $p_t^{\pi t}$ is the tth probability on the path $\boldsymbol{\pi}$. CTC calculates Eq. (7) effectively by using dynamic programming. We note that framewise decoding is performed only during training.

4.3 Decoding Process

The decoder predicts words sequentially using the output from the encoder and the input word sequence. The Word Embedding layer converts discrete numbers corresponding to words into 64-dimensional vectors. The subsequent processes are the same as those in the standard Transformer [6].

The sequence-to-sequence decoding is performed as follows: In the training phase, $\boldsymbol{Y} = \{y_1 = <\text{start}>, \ldots, y_s \in \boldsymbol{W}, \ldots, y_S = <\text{end}>, y_{S+1} = <\text{pad}>, \ldots, y_{S_{max}} = <\text{pad}>\}$ is inputted into the decoder. Similar to the encoder, the final FC and softmax layers convert the output of the Transformer decoder into the word probabilities $\boldsymbol{Q} = \{\boldsymbol{q}_s\}$; $\boldsymbol{q}_s \in [0.0, 1.0]^{\times|\boldsymbol{W}|+3}$. The three additional dimensions of $\boldsymbol{q}_s$ correspond to <start>, <end>, and <pad>.

In the test phase, $y_1 = <\text{start}>$ is input first, and then the model predicts the next word $\hat{y}_1$. The predicted word is concatenated as $\{y_1 = <\text{start}>, \ldots, y_s = \hat{y}_{s-1}\}$ and input into the next repetition. This process is repeated until $\hat{y}_s = <\text{end}>$ is outputted.

Let $\boldsymbol{Y}^*; \boldsymbol{y}_s^* \in \{0.0, 1.0\}^{\times|\boldsymbol{W}|+3}, \boldsymbol{y}_s \cdot \boldsymbol{e}^{\mathsf{T}} = 1.0$ be a one-hot vector representation of $\boldsymbol{Y}$, where $\boldsymbol{e} = \{1.0\}^{\times|\boldsymbol{W}|+3}$ is a unit vector. The sequence-to-sequence decoding loss based on CE is calculated as follows:

$$L_{CE} = -\frac{1}{S} \sum_{s} \boldsymbol{y}_s^* \cdot \log(\boldsymbol{q}_s^{\mathsf{T}}). \tag{8}$$

Our training strategy minimizes the weighted sum of the CTC loss L_{CTC} and CE loss L_{CE}. The weighted sum of the losses are defined as

$$L = \lambda L_{CTC} + L_{CE}, \tag{9}$$

where λ is a hyperparameter to control the relative importance of the CTC loss and is evaluated in Section 5.

5. Evaluation

5.1 Dataset

We built a sign language video dataset to evaluate the proposed method. The signers are 37 adults who have experience in sign language. We assumed a conversation at the city office and collected 275 types of isolated word videos and 120 types of sentence videos. **Table 1** shows examples of the sentences. We note that the translations in Table **1** are just to explain the content, and the sign translation task is beyond the scope of this paper. We recorded the videos with a smartphone camera. All the video frames were recorded at 30 frames per second and 640×360 pixels. The single signer sat on a chair and performed each word and sentence five times in front of the camera. The signers were posed in the static posture at the beginning and end of the sign, as shown in Fig.**1**. We defined an action instance as the frames between these static postures. We focus on tracking-based sign language recognition in this research. Therefore, we used the tracking points extracted by OpenPose as the inputs and discarded the raw video frames.

Table 2 shows a summary of the dataset. The number of action types is in parentheses. We note that horizontally flipped tracking sequences were added to avoid the effect of the dominant hand. The number of videos from each signer was not balanced in the dataset. Hence, we selected the two signers with the largest num-

Table 1 Examples of sentences

Words	[入籍 (registration of marriage)], [希望 (Hope)]
Translation	I'd like to register for the marriage.
Words	[市 (City)], [外 (Outside)], [両親 (Parents)], [住む (Live)]
Translation	My parents live outside of the city.
Words	[今度 (Next time)], [夫婦 (Couple)], [一緒に (Together)], [引っ越し (Move)]
Translation	We will move together next time.
Words	[子供 (Child)], [手当 (Allowance)], [期限 (Deadline)], [振り込まれる (Transfer)], [あなた (You)]
Translation	When will my children's allowance be transferred?
Words	[私 (I)], [今 (Now)], [月 (Month)], [60], [5], [歳 (Year)], [なる (become)], [終わり (Finish)]
Translation	I became 65 years old this month.

Table 2 Summary of sign language video dataset

Subset types	Training		Test	
# of signers	35		2	
# of isolated words	22640	(275)	3862	(210)
# of sentences	7466	(120)	1372	(105)

Table 3 Summary of the word lengths in the sentences

# of words	2	3	4	5	6	7	8	9
# of sentences	2	5	23	23	28	13	18	4
# of words	10	11						
# of sentences	3	1						

ber of videos recorded for the isolated words and sentences as the test signers. The sentences were composed of combinations of 200 types of words. 42 types of words in the sentences were not included in the isolated word videos. Therefore, the total vocabulary of the dataset was 317.

Table 3 shows a summary of the word lengths in the sentences. The most frequently occurring sentences were short sentences of a few words, and the maximum word length of the dataset was eleven.

5.2 Training Details

The input feature sequences and word sequences are required to have a fixed length during training, as mentioned in Section 4.1. The maximum number of frames and words were 578 and 11, respectively. Therefore, the input feature sequences and word sequences were padded with $T_{max} = 578$ and $S_{max} = 13$, respectively. We note that the input word sequences includes <start > and <end >. The lengths of the input feature sequences are known in the experiment. Sign detection techniques [34] are available when the lengths of input feature sequences are unknown.

The training settings were as follows: The batch size was 32 throughout the training. We used the adaptive moment estimation method to update the parameters. We repeated the training loop 150 times.

5.3 Recognition Performance

We used the word error rate (WER) as the performance metric in this study. The WER is defined as

$$WER = \frac{\mathrm{dist}(L_1, L_2)}{\max(|L_1|, |L_2|)} * 100, \tag{10}$$

Table 4 Performance summary

Models	λ	Isolated [%]	Sentence [%]
GRU-Attention	0.0	34.35	12.30
Transformer	0.0	28.23	8.90
STGC-Transformer	0.0	21.08	5.19
	1.0	36.28	16.39
	0.1	45.29	23.15
	0.01	**12.14**	**2.07**
	0.001	16.49	8.32

where $\mathrm{dist}(\cdot, \cdot)$ is the Levenshtein distance between two sequences, and $|L|$ indicates the word length.

Table 4 summarizes the minimum average WERs during training. Table **4** includes GRU/Attention [20] and naive Transformer in which the STGC layers were absent for comparison. We applied hyperbolic tangent in the first FC layer of the encoder in the naive Transformer because this achieved a better result in our evaluation. $\lambda = 0.0$ implies that only the CE loss was applied. As described in Table **4**, STGC-Transformer with single-task learning achieved better performance than GRU/attention and the naive Transformer for both isolated words and sentences. STGC-Transformer improved the WERs by 13.27% and 7.15% compared to GRU/attention and naive Transformer for isolated words, respectively. Similarly, the proposed method improved the WERs by 7.11% and 3.71% for sentences, respectively. A remarkable improvement was achieved in isolated words, which indicates the effectiveness of feature extraction by STGC. Moreover, the multi-task learning improved the WERs of the proposed method by 8.94% and 3.12% for the isolated words and sentences, respectively, when $\lambda = 0.01$ was applied. These improvements show the effectiveness of the proposed training strategy. All models achieved good WERs in sentences. It could be inferred that additive information, such as the length of the input sequence and the decoder's language model, contributed to improving the results.

Figure 5 shows the behaviors of the attention weights of the second attention block in the decoder. The video shown in Fig. **1**(c) was used to generate Fig. **5**. The horizontal and vertical axes indicate the frame indices and attention weights, respectively. The blue and orange lines represent the attention weights for the prediction of the first and second words, respectively. The images in Fig.**5**

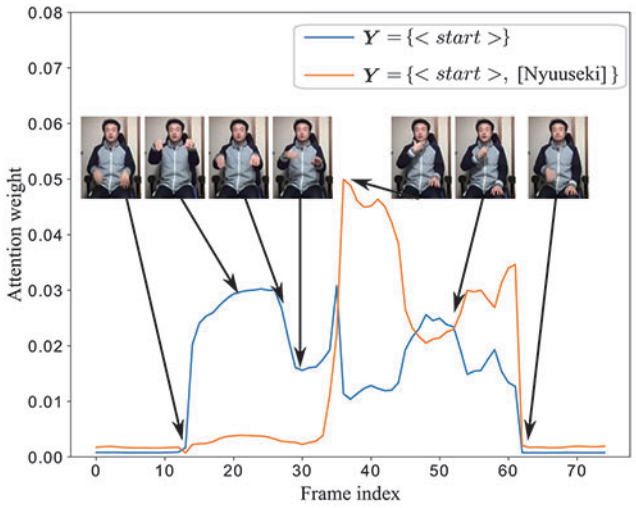
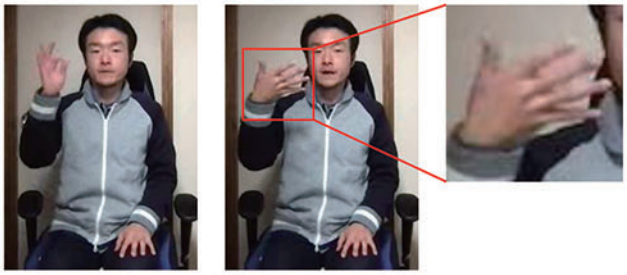
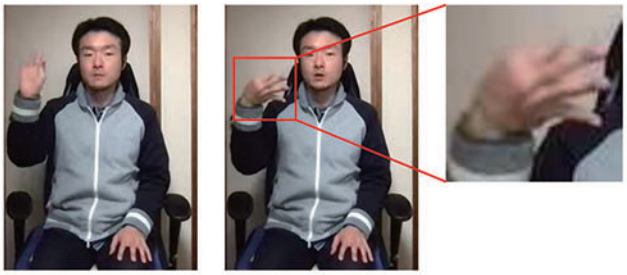


Fig. 5 Behaviors of the attention weights



(a) Isolated word "Zei"



(b) Isolated word "Mono"

Fig. 6 Example of similar sign motions

show the motions corresponding to the frame indices. The arrows indicate the correspondences. The attention layer gave higher and lower weights to the word motions and marginal motions, respectively.

Finally, we show an example of a failed case in isolated word recognition. **Figure 6 (a)** and (b) are examples of the words "税 (Zei)" and "物 (Mono)," which mean "tax" and "object" respectively. For the word "Zei," a ring is first created with the thumb and index finger, and then the hand is rotated while opening the ring. For the word "Mono," a ring is first created with the thumb and index finger, and then the hand is rotated while maintaining the shape of the ring. The motions of the two words are very similar, except for the final hand shape. The tracking accuracy of the current OpenPose system may not be sufficient to capture this fine difference.

6. Conclusions and Future Work

We proposed STGC-Transformer for tracking-based sign language recognition. We evaluated the proposed method using a dataset that included 275 types of isolated word videos and 120 types of sequence videos by 37 signers. In the evaluation, STGC-Transformer with single-task learning achieved 21.08% and 5.19% WERs for isolated words and sentences, respectively. Similarly, STGC-Transformer with multi-task learning achieved 12.14% and 2.07% WERs for isolated words and sentences, respectively.

Interestingly, the accuracy of continuous word recognition was superior to that of isolated word recognition. The variation of the input sequence lengths and language model of the decoder contributed to the recognition. However, these contributions were small in isolated word recognition. We assume that the tracking and feature extraction performance has more significant effects on isolated word recognition.

Spatial-temporal graph convolution[5] is a hot topic in action recognition, and its extensions[25)–27)] have been actively researched. These methods can be expected to improve the proposed method.

Although the dataset used in this work includes both isolated words and sentences, it does not have enough vocabulary and sentence patterns. The extension of the dataset must be addressed in future research.

Acknowledgements

This research is supported by SoftBank Corp.

References

1) O. Koller, H. Ney, and R. Bowden: Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2016) 3793.
2) O. Koller, S. Zargaran, and H. Ney: Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent cnn-hmms, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2017) 3416.
3) N.C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden: Neural sign language translation, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2018) 7784.
4) N.C. Camgoz, O. Koller, S. Hadfield, and R. Bowden: Sign language transformers: Joint end-to-end sign language recognition and translation, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2020) 10023.
5) S. Yan, Y. Xiong, and D. Lin: Spatial temporal graph convolutional networks for skeleton-based action recognition, Proc. 32nd AAAI Conference on Artificial Intelligence, (2018) 7444.
6) A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin: Attention is all you need, Advances in Neural Information Processing Systems, **30**, (2017) 5998.
7) A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks, Proc. the 23rd International Conference on Machine Learning, (2006) 369.
8) O. Koller, N.C. Camgoz, H. Ney, and R. Bowden: Weakly supervised learning with multi-stream cnn-lstm-hmms to discover sequential parallelism in sign language videos, IEEE Transactions on Pattern Analysis and Machine Intelligence, **42**, 9, (2020) 2306.
9) D. Bahdanau, K. Cho, and Y. Bengio: Neural machine translation by jointly learning to align and translate, Proc. Third International Conference on Learning Representations, (2015) 1.
10) H. Cooper, N. Pugeault, and R. Bowden: Reading the signs: A video based sign dictionary, Proc. IEEE International Conference on Computer Vision Workshops, (2011) 914.
11) P. Tomas, C. James, and Z. Andrew: Large-scale learning of sign language by watching tv (using co-occurrences), Proc. British Machine

Vision Conference, (2013) 1.
12) J. Forster, O. Koller, C. Oberdörfer, Y. Gweth, and H. Ney: Improving continuous sign language recognition: Speech recognition techniques and system design, Proc. Fourth Workshop on Speech and Language Processing for Assistive Technologies, (2013) 41.
13) D. Li, C. Rodriguez, X. Yu, and H. Li: Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison, Proc. IEEE/CVF Winter Conference on Applications of Computer Vision, (2020) 1459.
14) M. De Coster, M. Van Herreweghe, and J. Dambre: Sign language recognition with transformer networks, Proc. 12th Language Resources and Evaluation Conference, (2020) 6018.
15) Z. Cao, T. Simon, S. Wei, and Y. Sheikh: Realtime multi-person 2d pose estimation using part affinity fields, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2017) 7291.
16) J. Wang, Z. Liu, Y. Wu, and J. Yuan: Mining actionlet ensemble for action recognition with depth cameras, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2012) 1290.
17) M.E. Hussein, M. Torki, M.A. Gowayyed, and M. El-Saban: Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations, Proc. 23rd International Joint Conference on Artificial Intelligence, (2013) 2466.
18) R. Vemulapalli, F. Arrate, and R. Chellappa: Human action recognition by representing 3d skeletons as points in a lie group, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2014) 588.
19) F. Angelini, Z. Fu, Y. Long, L. Shao, and S.M. Naqvi: 2d pose-based real-time human action recognition with occlusion-handling, IEEE Transactions on Multimedia, **22**, 6, (2020) 1433.
20) N. Takayama and T. Hiroki: Data augmentation using feature interpolation of individual words for compound word recognition of sign language, Proc. International Conference on Cyberworlds, (2020) 137.
21) A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang: Ntu rgb+d: A large scale dataset for 3d human activity analysis, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2016) 1010.
22) W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie: Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks, Proc. 30th AAAI Conference on Artificial Intelligence, (2016) 3697.
23) Y. Du, Y. Fu, and L. Wang: Skeleton based action recognition with convolutional neural network, Proc. Third IAPR Asian Conference on Pattern Recognition, (2015) 579.
24) Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid: A new representation of skeleton sequences for 3d action recognition, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2017) 4570.
25) K. Thakkar and P.J. Narayanan: Part-based graph convolutional network for action recognition, Proc. British Machine Vision Conference, (2018) 270:1.
26) F. Li, A. Zhu, Y. Xu, R. Cui, and G. Hua: Multi-stream and enhanced spatial-temporal graph convolution network for skeleton-based action recognition, IEEE Access, **8**, (2020) 97757.
27) L. Shi, Y. Zhang, J. Cheng, and H. Lu: Skeleton-based action recognition with multi-stream adaptive graph convolutional networks, IEEE Transactions on Image Processing, **29**, (2020) 9532.
28) R. Caruana: Multitask learning: A knowledge-based source of inductive bias, Proc. the Tenth International Conference on Machine Learning, (1993) 41.
29) R. Cui, H. Liu, and C. Zhang: Recurrent convolutional neural networks for continuous sign language recognition by staged optimization, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (2017) 1610.
30) J. Pu, W. Zhou, and H. Li: Iterative alignment network for continuous sign language recognition, Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, (2019) 4160.
31) M. Zhou, M. Ng, Z. Cai, and C.C. Ka: Self-attention-based fully-inception networks for continuous sign language recognition, Proc. the 24th European Conference on Artificial Intelligence, **325**, (2020) 2832.
32) I. Papastratis, K. Dimitropoulos, D. Konstantinidis, and P. Daras: Continuous sign language recognition through cross-modal alignment of video and text embeddings in a joint-latent space, IEEE Access, **8**, (2020) 91170.
33) H. Li, L. Gao, R. Han, L. Wan, and W. Feng: Key action and joint ctc-attention based sign language recognition, Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, (2020) 2348.
34) A. Moryossef, I. Tsochantaridis, R. Aharoni, S. Ebling, and S. Narayanan: Real-time sign language detection using human pose estimation, Proc. European Conference on Computer Vision Workshops, LNCS 12536, (2020) 237.