SimulSLT: End-to-End Simultaneous Sign Language Translation

Aoxiong Yin¹, Zhou Zhao*, Jinglin Liu¹, Weike Jin¹, Meng Zhang², Xingshan Zeng², Xiaofei He¹

¹Zhejiang University
²Huawei Noah’s Ark Lab

{yinaoxiong,zhozhou,jinglinliu,weikejin}@zju.edu.cn, zhangmeng92@huawei.com, zxshamson@gmail.com, xiaofei_h@qq.com

ABSTRACT

Sign language translation as a kind of technology with profound social significance has attracted growing researchers’ interest in recent years. However, the existing sign language translation methods need to read all the videos before starting the translation, which leads to a high inference latency and also limits their application in real-life scenarios. To solve this problem, we propose SimulSLT, the first end-to-end simultaneous sign language translation model, which can translate sign language videos into target text concurrently. SimulSLT is composed of a text decoder, a boundary predictor, and a masked encoder. We 1) use the wait-k strategy for simultaneous translation. 2) design a novel boundary predictor based on the integrate-and-fire module to output the gloss boundary, which is used to model the correspondence between the sign language video and the gloss. 3) propose an innovative re-encode method to help the model obtain more abundant contextual information, which allows the existing video features to interact fully. The experimental results conducted on the RWTH-PHOENIX-Weather 2014T dataset show that SimulSLT achieves BLEU scores that exceed the latest end-to-end non-simultaneous sign language translation model while maintaining low latency, which proves the effectiveness of our method.

1 INTRODUCTION

Sign language is a visual language widely used by an estimated 466 million deaf or hard-of-hearing people [41]. They use various methods to convey information, such as gestures, movements, mouth shapes, facial expressions, etc [39]. However, there are certain obstacles for normal people to understand this language, which causes inconvenience to their daily lives. This motivates researchers to design the AI sign language translation system [27, 28, 52].

Previous studies on sign language translation (SLT) mainly focus on non-simultaneous translation methods [5–7, 14, 24, 32, 51]. The model needs to read an entire video before starting translation. Therefore, if they are applied to simultaneous scenes in actual use, such as assisting deaf or hard-of-hearing people to seek help from staff in public places like airports, they will all suffer high inference delays without exception. This is unacceptable for practical applications.

Therefore, we need a simultaneous sign language translation system that can better balance translation quality and delay to meet the needs of practical applications. Simultaneous sign language translation is a very challenging task. For example in the simultaneous translation task, the model needs to decide when to switch from the encoding state to the decoding state. In simultaneous text translation, this decision is made at the token (word or BPE) level [36]. However, in this task, since there is no clear semantic unit and its corresponding boundary, we need to design a suitable boundary predictor, to make this decision.

Hence, we propose the SimulSLT model, an end-to-end simultaneous sign language translation model based on Transformer. The main model is mainly composed of three parts, 1) a masked encoder for encoding streaming input sign language video; 2) a text decoder where the cross-attention follows the wait-k strategy [35]; 3) a boundary predictor for predicting the boundary of gloss words. An example to illustrate how our model works is shown in Figure 1. First, the sign language video is divided into segments corresponding to gloss by the boundary predictor. Next, after waiting for the video frames corresponding to k (here is 2) glosses, the model starts...
to translate the target text, and then translates a text every time it encounters a word boundary. Until all the videos have been read, the model enters the non-simultaneous translation state. To get the video segment corresponding to the gloss, we design a novel boundary predictor based on the integrate-and-fire mechanism, which works similarly to human neurons. The stimulus signal generated by the streamed sign language video is continuously accumulated in the boundary predictor until the signal exceeds the threshold value of the boundary predictor to fire the word boundary, then the signal falls back, and the boundary predictor enters the next round of accumulation.

However, due to the lack of available alignment annotation information, we cannot directly train the boundary predictor. To solve this problem, we introduce an auxiliary gloss decoder to help the boundary predictor learn alignment information.

In terms of context feature acquisition, most of the previous work directly used the output of the encoder before the boundary predictor as the context feature for text decoding. In this case, each video frame can only interact with the video before it, which makes the interaction between known video information during decoding insufficient. Therefore, we design an innovative re-encode method to let the output of the encoder contain more abundant contextual information, which can make the obtained video frame information fully interact with each other while ensuring that the algorithm time complexity is kept at $O(n^2)$.

Considering the difficulty of simultaneous sign language translation tasks, we introduce a knowledge distillation method to help model optimization, which is used to transfer knowledge from non-simultaneous teacher models to SimulSLT. In addition, we also add an auxiliary connectionist temporal classification (CTC) decoder behind the encoder to enhance the feature extraction capability of the encoder and help the boundary predictor to better learn alignment information. The experimental results show that on the RWTH-PHOENIX-Weather 2014T (PHOENIX14T) dataset [5], SimulSLT even obtains BLEU scores that exceed the latest end-to-end non-simultaneous sign language translation model while maintaining low latency, which proves the effectiveness of our method. We also conduct ablation experiments to verify the effectiveness of all proposed methods in SimulSLT.

To summarize, the contributions of this work are as follows:

- We are the first to explore the simultaneous sign language translation problem, and we propose SimulSLT, which is a simultaneous sign language translation model based on wait-k strategy.
- We design a novel boundary predictor to help the model decide when to switch from the encoding state to the decoding state. It efficiently learns the correspondence between sign language video and gloss.
- We design a general re-encode to help the simultaneous translation model to obtain more abundant contextual information.
- Extensive experiments are conducted on the public benchmark dataset (PHOENIX14T) and the results show that SimulSLT achieves translation accuracy higher than the latest end-to-end non-simultaneous sign language translation methods while maintaining low latency. A broad range of new baseline results can guide future research in this field.

2 RELATED WORKS

Research into sign language translation has a long history [20, 21, 44, 46, 47]. In recent years, with the rise of deep learning, many people have tried to use neural network methods to deal with SLT tasks and achieved good results. Due to lack of data, early research mainly focused on Isolated Sign Language Recognition [16–18, 29, 30, 50]. In recent years, with the emergence of a series of high-quality datasets [5, 19, 25], researchers have begun to turn to the study of continuous sign language recognition (CSLR) and sign language translation (SLT). The CSLR task aims to convert the sign language video into the corresponding sign language gloss, and these two sequences have the same order. However, the equivalent spoken language that can be understood by ordinary people is different in length and order compared with gloss. There is a more detailed introduction about the difference between CSLR and SLT in [5].

Sign language translation aims to translate a continuous sign language video into the corresponding spoken language. Camgoz et al. [5] first formalize the sign language translation task in the framework of neural machine translation (NMT) and released the first publicly available sign language translation dataset, RWTH-PHOENIX-Weather-2014T (PHOENIX14T). Soon after, They use the Transformer structure to design an end-to-end translation model using gloss and text as supervision signals, instead of relying on the CSLR model and using gloss as an intermediate representation [7]. This work shows that using gloss as an intermediate language is not a good choice. Compared with the two-stage model, the end-to-end video to text model can better perform model training and information transmission. The work of Yin and Read [51] further proves this point. They obtain better performance by using a STMC network for tokenization instead of translating GT glosses.

The previous work has used gloss annotations as auxiliary supervision signals [5, 7, 24, 45, 51]. However, the acquisition of gloss annotations is expensive and difficult since it requires sign language experts to annotate. Therefore, it is necessary to explore how to train the model when the gloss label is missing. Li et al. [32] reduced the model’s dependence on gloss annotation by fine-tuning the feature extractor on the word-level sign language dataset of another different sign language. This shows that different sign languages have a lot in common in the underlying features. The work of Or-bay and Akarun [40] showed that good hand shape representations could improve translation performance, which is consistent with our common sense because sign language usually conveys much information through gestures.

Delay is also essential for sign language translation. However, the research that has been conducted in SLT to date have to read a complete sign language video to start the translation, which will cause a serious out of sync between the signer and the text generated by the model. Therefore, we are the first to explore the simultaneous sign language translation problem. Research on simultaneous translation in other fields has made some progress [1, 35, 37]. Ma et al. [35] present a very simple yet effective “wait-k” policy for simultaneous text translation model. The model will first
wait k source words and then translates concurrently with the rest of source sentence. By applying “wait-k” policy to the field of simultaneous speech translation and combining multi-task learning and knowledge distillation methods, Ren et al. [43] proposed the SimulSpeech model. Their work proves that the end-to-end simultaneous translation system has lower latency and higher performance than the cascaded translation system.

The integrate-and-fire model is the most prominent spiking neuron model [4, 38]. It simulates the activity of neurons in biology. It increases the membrane potential according to the input of the neuron and fires after reaching the threshold. After firing, the membrane potential drops to a lower level and starts the next round of accumulation. It can model the relationship between continuous and discrete sequences very well. Recently, the integrate-and-fire module has been applied to speech recognition and lip recognition tasks and achieved good results [11, 33].

3 METHODS

3.1 Overview

In this section, we introduce SimulSLT and describe our methods thoroughly. As shown in Figure 2(a), the main model of SimulSLT is based on the Transformer [49] architecture and consists of a CNN video feature extractor, a masked encoder, a boundary predictor, and a wait-k decoder. In order to help the boundary predictor learn the alignment information better, we introduce an auxiliary gloss decoder and an auxiliary connectionist temporal classification (CTC) decoder. We designed a new re-encoding method to enhance the feature expression ability of the encoder and help the model obtain more abundant contextual semantic information. Besides, to further tackle the challenges in SimulSLT, we propose a knowledge distillation method to reduce the optimization difficulty of the model and improve the performance of the model. We will introduce them in detail in the following subsections.

3.2 Masked Encoder

As shown in Figure 2(b), the encoder in SimulSLT consists of a stacked masked self-attention layer and a feedforward layer. In order to adapt to the task of simultaneous translation, we designed a masked self-attention mechanism to ensure that the current frame can only see the video frame before it. As shown in Figure 2(b), the streaming sign language video features are first input to the encoder on the left, which is used to learn the alignment information between the video and the gloss, and provide the encoded features for the boundary predictor. We follow the boundary predictor with an auxiliary gloss decoder, which is used to provide alignment supervision information between the video and the gloss during training. A connectionist temporal classification (CTC) [13] decoder with CTC loss is also added to the encoder to learn more meaningful spatio-temporal representations of the sign. Whenever the wait-k strategy decides to switch to the decoding state and the video stream has not ended, as shown on the right in Figure 2(b), we apply the re-encode method to re-encode the video features that have been obtained. This method will be introduced in detail in section 3.5. After that, we further merge the outputs of the two encoders and output them to the text decoder for decoding (here we use the direct addition method for fusion). The two encoders share the same weight.

3.3 Boundary Predictor

We build our boundary predictor based on the Integrate-and-fire module. Like natural neurons, it emits pulses after signal accumulation reaches a threshold, and each pulse represents a word boundary. The encoder’s output sequence \( H = (h_1, h_2, ..., h_n) \) first passes through a multi-layer perceptron (MLP) to obtain the weight embedding sequence \( W = (w_1, w_2, ..., w_n) \). The running process of the entire boundary predictor can be expressed by the following formula:

\[
\text{Boundary predictor} = \sum_{i=1}^{n} w_i h_i
\]
where formula 1 represents the calculation method of the MLP we designed, \(w\) represents the continuous stimulus signal in the time series, and a kind of technology module accumulates it and emits the word boundary \(b_j\) and gloss embedding \(g_j\) after reaching the threshold \(T\) (we set it to 1.0). When the accumulated stimulus signal reaches the threshold, the stimulus signal \(w_b\) will be split into two parts. The part one \(l_j\) within the threshold is used to calculate gloss embedding \(e_j\), and the remaining part \(r_j\) is used for the next accumulation to calculate \(e_{j+1}\). As shown in formula 5, gloss embedding is obtained by multiplying the weight within the threshold by encoder output and then adding them.

### 3.4 Decoder

#### 3.4.1 Auxiliary Decoder.** The auxiliary decoder is a decoder made up of multiple layers of Transformer decoders [49] with the cross-attention mechanism removed because we already have an a kind of technology module to align source and target. The decoder takes in the gloss embedding sequence \(E = (e_1, e_2, ..., e_n)\), and then generates the gloss tokens \(G = (g_1, g_2, ..., g_n)\). The decoding process only occurs during the training process for learning alignment information.

#### 3.4.2 Wait-k Decoder.** In the SimulSLT model, we adopt the wait-k strategy [35] for simultaneous translation. Let \((x, y)\) be a pair of video-text sequences. Given the source and target context, the model need to compute the distribution over the next target token.

\[
P(y_t | y_{<t}, x_{<t+k}; \theta)\tag{6}
\]

where \(\theta\) is the model parameter, \(y_{<t}\) represents the target tokens before position \(t\), and \(x_{<t+k}\) represents the video segments before position \(t+k\). To ensure that the decoder can only see \(t+k-1\) video segments when generating the target token \(y_t\), we generate a mask based on the word boundaries obtained by the previous prediction to mask the encoder’s output when performing cross-attention.

### 3.5 Re-encode Strategy

To simulate the real simultaneous translation task, most of the previous work used the lower triangular mask as shown in Figure 3(a) to ensure that the current frame can only see the frame before it, which is reasonable when performing word boundary prediction. However, directly using the encoder’s output in the boundary prediction stage to decode the text will bring some performance loss because each frame only performs self-attention calculation with the frame before it, and there is no sufficient interaction between the known frames.

An ideal re-encode method is shown in Figure 3(c). Whenever we predict a word boundary, we re-encode all the previous frames until the end of the sequence. The calculation of the self-attention part of this algorithm is as follows:

\[
\hat{e}_j = \frac{1}{k} \sum_k \beta_k v_k, \text{ where } \beta_k = \frac{\text{exp}(\text{sim}(v_j, v_k))}{\sum_q \text{exp}(\text{sim}(v_j, v_q))} \tag{7}
\]

\(j \leq b_i\)

where \(V = (v_1, v_2, ..., v_n)\) represents the input sequence of self-attention, \(b_i\) represents the position of the i-th word boundary. \(\hat{e}_j\) represents the value obtained after self-attention, and the sim function uses the dot product function after the dimension scaling mentioned in [49]. Let us consider the worst case that this algorithm can encounter: a word boundary is predicted for each frame. In this case, its calculation time is as follows:

\[
1^2 + 2^2 + ... + n^2 = \frac{n(n + 1)(2n + 1)}{6} \tag{8}
\]

Obviously the time complexity of the algorithm is \(O(n^3)\), while the time complexity of the original algorithm is \(O(n^2)\). At the same time, the algorithm requires re-encoding every time, which is not conducive to parallel training of the model. So we proposed an algorithm that only re-encode once as shown in Figure 3(b). The self-attention calculation method of the improved algorithm is as follows:

\[
\hat{e}_j = \frac{1}{k} \sum_k \beta_k v_k, \text{ where } \beta_k = \frac{\text{exp}(\text{sim}(v_j, v_k))}{\sum_q \text{exp}(\text{sim}(v_j, v_q))} \tag{9}
\]

\(b_{i-1} < j \leq b_i\)

In this algorithm, we only re-encode the frames in a word range, and the frames in the previous word range no longer calculate self-attention with the following frames. In this way, we can reduce the algorithm’s time complexity to \(O(n^2)\) without affecting the expression of word information.

In addition, since it only needs to re-encode once to get all the output needed for decoding, this allows us to train the model efficiently and in parallel. Our subsequent experimental results prove...
where this recoding strategy helps to improve the performance of the model.

### 3.6 Knowledge Distillation

Knowledge distillation [15, 23] is widely used to reduce the model’s delay and improve the performance of the student model. In our work, we use this method to transfer knowledge from a non-simultaneous teacher model to a SimulSLT model. First, we input the source video \( x \) into the teacher model to get logit \( z \). Then input \( z \) into the softmax-T function, which is used to construct a soft target \( y' \) to obtain information from the negative label.

\[
y' = \frac{\exp(z_j/T)}{\sum_j \exp(z_j/T)} \tag{10}
\]

where \( y' \) the soft target which is used to supervise the training of the student model.

### 3.7 Training Of SimulSLT

In The SimulSLT model, the CTC decoder is used to enhance the expressive ability of the encoder, and we use CTC loss to optimize it. CTC introduces a set of intermediate paths \( \phi(y) \) called CTC path for a target text sequence \( y \). Multiple CTC paths may correspond to the same target text sequence, because the length of the source sequence is generally much longer than the target sequence. The probability of the target sequence is the sum of the probabilities of all corresponding intermediate paths:

\[
P_{ctc}(y|x) = \sum_{c \in \phi(y)} P(c|\phi(y)) \tag{11}
\]

then, the CTC loss is formulated as:

\[
\mathcal{L}_{ctc} = -\sum_{(x, y) \in (X, Y_{\text{gloss}})} \sum_{c \in \phi(y)} P(c|\phi(y)) \tag{12}
\]

where \((X, Y_{\text{gloss}})\) denotes the set of source video and target gloss pairs in the data.

We use cross entropy loss and length loss to optimize the auxiliary decoder and a kind of technology module, which can be formulated as:

\[
\mathcal{L}_{IF} = -\sum_{(x, y) \in (X, Y_{\text{gloss}})} \log(P_{IF}(y|x)) + (\hat{S}_x - S_x)^2 \tag{13}
\]

where \( S_x \) is the length of the target gloss, and \( \hat{S}_x \) is the sum of all weight embedding \( W \), which represents the length of the prediction sequence.

We also use cross-entropy loss to optimize the main translation task of SimulSLT:

\[
\mathcal{L}_{hard} = -\sum_{(x, y) \in (X, Y)} \log(P(y|x)) \tag{14}
\]

where \( (X, Y) \) denotes the set of video-text sequence pairs.

Therefore, the total loss function to train the SimulSLT model is:

\[
\mathcal{L} = \lambda_1 \mathcal{L}_{ctc} + \lambda_2 \mathcal{L}_{IF} + \lambda_3 \mathcal{L}_{soft} + \lambda_4 \mathcal{L}_{hard} \tag{15}
\]

where the \( \lambda_1 \) to \( \lambda_4 \) are hyperparameters to trade off these losses, and the calculation method of \( \mathcal{L}_{soft} \) is the same as \( \mathcal{L}_{\text{WORD–KD}} \) proposed by Kim and Rush [23].

### 4 EXPERIMENTS

#### 4.1 Datasets

We evaluate the SimulSLT model on the RWTH-PHOENIX-Weather 2014T (PHOENIX14T) dataset [5], which is the most commonly used large-scale SLT dataset. Its data is collected from the weather forecast of the German public television station PHOENIX, including parallel sign language videos, gloss annotations, and corresponding text translations. We follow the official data set partitioning protocol, where the training set, validation set, and test set contain 7096, 519, and 642 samples, respectively. The dataset contains consecutive sign language videos from 9 different signers with a vocabulary of 1066 different signs. The text annotations in the dataset are German spoken language with a vocabulary of 2887 different words.

#### 4.2 Implementation Details

Our model’s main structure is the Transformer [49], which has been widely used in machine translation. The number of hidden units in the model, the number of heads, the number of encoder and decoder layers are set to 512, 8, 3, 3, respectively. We also use dropout with 0.3 and 0.6 drop rates on encoder and decoder layers to mitigate overfitting. We use the word embeddings trained by FastText [3] on the Wikipedia German dataset as the initial word embeddings of the model, and the word embeddings of gloss are initialized randomly. We adopt Xavier initialization [12] to initialize our network.

#### 4.3 Training Setup

We train our model on a single Nvidia 2080ti GPU with a total batch size of 32. We use the Adam optimizer with a learning rate of \( 5 \times 10^{-4} (\beta_1=0.9, \beta_2=0.998) \), and the weight decays to \( 10^{-3} \). We utilize the plateau learning rate schedule to update the learning rate, which tracks the BLEU [42] score on the validation set, the patience and factor are set to 9 and 0.5, respectively. The validation set is evaluated every 100 steps. During validation, we use a beam search algorithm with a beam size of 3 and a length penalty value -1 to decode the text sequence. Training ends when the learning rate is less than \( 10^{-7} \). During training, the weights of the loss function \( \lambda_1 - \lambda_4 \) are set to 10, 1, 0.6, and 0.4 respectively.

#### 4.4 Evaluation Metrics

We use the BLEU [42] and ROUGE-L [10] scores to evaluate the quality of the translation. We use average lagging (AL) [36] and average proportion (AP) [8] to evaluate the latency of our model. The former is used to measure the degree of out of sync between the model output and the signer, and the latter is used to measure the average absolute delay cost of each target text. Assuming that the input video sequence of the model is \( X = (x_1, x_2, ..., x_n) \), the reference target text sequence is \( Y^* = (y_1^*, y_2^*, ..., y_n^*) \), and the predicted target text sequence is \( Y = (y_1, y_2, ..., y_n) \). The calculation
formula of AL is as follows:

\[
AL = \frac{1}{\tau(X)} \sum_{i=1}^{\tau(X)} d(y_i) - \frac{|X|}{|Y|} \cdot T_s \cdot (i - 1) \tag{16}
\]

where \(T_s\) represents the time interval between obtaining two video frames, \(|Y^*|\) is the length of the reference translation text, \(|X|\) is the length of the input video, and \(\tau(X)\) is the index of the first target token generated when the model has read all input videos. \(d(y_i)\) represents the time it takes to generate \(y_i\), \(d(y_i)\) is equal to \(T_s\) multiplied by the number of video frames that has been read when \(y_i\) is generated. The calculation formula of \(AP\) is as follows:

\[
AP = \frac{1}{T_s \cdot |X| \cdot |Y|} \sum_{i=1}^{|Y|} t(i) \tag{17}
\]

where, \(t(i)\) represents the duration of the model from the beginning to the calculation of the \(i\)-th target text.

4.5 Experimental Results

4.5.1 Translation Accuracy. In this section, we compare the performance of SimulSLT with the existing end-to-end non-simultaneous sign language translation model and explore the impact of different \(k\) settings on the model performance. The comparison results are shown in Table 1. In order to provide a broad baseline for future research in this field, we also tested the SimulSLT model trained without gloss annotations. In this case, we simply use \(p = \text{mean} \left( \frac{2M}{N} \right)\) to segment the video statically, where \(v_1\) represents the length of the video and \(q_1\) represents the length of the gloss. In addition, the native modal in Table 3 also uses this video segmentation method.

According to the different data types used when training the model, we divided the comparison results into three groups. All models of the first group are trained with all the data set information, including gloss annotations (here we use the same CNN network as [7] as the feature extractor.) Although the models in the second group did not use gloss during training, they all used other additional data related to sign language. For example, TSPNet-Joint [32] uses the additional American Sign Language video and its corresponding gloss information provided in the two data sets [22, 31], and both SLT-Multitask [40] and SimulSLT use the additional gesture information provided in the dataset [26]. We use the pre-trained CNN network in [26] to extract hand shape features as the input of the model. All models in the third group are not trained with gloss annotations, and the CNN network (we use EfficientNet [48]) pre-trained on ImageNet [9] is used as the feature extractor. We did not list the model in [51] in the table because it is a two-stage model of sign2gloss2text. As shown in the table, SimulSLT with \(k\) set to 7 achieves the best performance compared to other non-simultaneous sign language translation models in the three groups. These results show that our model still has excellent performance under the condition of lower latency.

Table 2 shows the BLEU scores of SimulSLT trained with different \(k\). We can sacrifice a part of the performance to obtain lower latency by setting a smaller \(k\).

4.5.2 Translation Delay. In this section, we explore the relationship between translation delay and translation quality, and compare our model with the most advanced end-to-end non-simultaneous sign language translation model. We plot the translation quality (denoted in BLEU scores) against the latency metrics (AL) of SimulSLT models trained on different data in Figure 4(a). We can see that the performance of the model gradually improves as \(k\) becomes larger, but the translation delay is gradually becoming larger. Therefore, in actual application, we can set \(k\) according to project requirements to achieve a balance between quality and delay.

The dotted line in the figure shows the comparison between our model and the non-simultaneous model. We can find that SimulSLT surpasses the non-simultaneous model in terms of translation performance and translation delay. Especially when no gloss is used, even when \(k\) is set to 1, SimulSLT is higher than the most advanced non-simultaneous model. In terms of latency, SimulSLT can achieve the same performance as the non-simultaneous model in almost half the time.

| Method                  | DEV | TEST |
|-------------------------|-----|------|
|                         | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | ROUGE |
| Joint-SLT [7]           | 47.26 | 34.40 | 27.05 | 22.38 | -     |
| SimulSLT (ours)         | 47.76 | 35.33 | 27.85 | 22.85 | 49.21 |
| SLT-Multitask [40]      | inf   | -     | -     | -     | -     |
| TSPNet-Joint [32]       | inf   | -     | -     | -     | -     |
| SimulSLT (ours)         | 36.21 | 23.88 | 17.41 | 13.57 | 36.38 |
| Conv2d-RNN(L) [5, 34]  | 31.58 | 18.98 | 13.22 | 10.00 | 32.60 |
| Conv2d-RNN(B) [2, 5]   | 31.87 | 19.11 | 13.16 | 9.94  | 31.80 |
| SimulSLT (ours)         | 36.01 | 22.60 | 16.05 | 12.39 | 36.04 |

Table 2: The BLEU scores of SimulSLT trained with different data on the test dataset of PHOENIX14T.

| model                  | k=1 | k=3 | k=5 | k=7 |
|------------------------|-----|-----|-----|-----|
| SimulSLT(with gloss)   | 19.27 | 21.62 | 22.58 | 23.14 |
| SimulSLT(additional data) | 12.66 | 12.83 | 13.46 | 14.10 |
| SimulSLT(without gloss)| 11.24 | 11.37 | 11.91 | 12.27 |
4.6 Ablation Study

In this subsection, we will introduce the results of our ablation experiments on the PHOENIX14T dataset, and analyze the effectiveness of our proposed method through the experimental results. The experimental results in the case of using gloss annotations are shown in Table 3. We can find that our proposed method can improve the translation accuracy of different wait-k.

### 4.6.1 The Effectiveness of Boundary Predictor.

As shown in Table 3, since the video segment corresponding to a gloss in sign language is not of a fixed length, simply using a fixed segmentation will cause information misalignment and insufficient information obtained during decoding. By adding a boundary predictor, the model can better learn the correspondence between the video and the gloss, and generate a more accurate boundary. From the BLEU score in the second row, it can be seen that the translation accuracy with different wait-k can be improved by adding a boundary predictor to Native SLT model.

### 4.6.2 The Effectiveness of CTC.

The result in the third row shows that the translation accuracy of the model can be further improved by adding an auxiliary connectionist temporal classification (CTC) decoder. Through the supervision of CTC loss, the encoder can have more robust feature expression ability, and the boundary predictor can also learn the alignment information better.

### 4.6.3 The Effectiveness of Knowledge Distillation.

We further explored the impact of knowledge distillation on model performance(show Row 4 vs Row 2). Knowledge distillation can transfer the knowledge learned by the teacher model to the student model, which reduces the difficulty of optimizing the student model. The results show that knowledge distillation at the data level makes the model achieve a greater performance improvement. We also tested the impact of knowledge distillation on the performance of the model in the
other two cases, and the results are shown in Figure 5. It can be
seen that by adding knowledge distillation, the model has achieved
an improvement in the translation accuracy in the different wait-k
settings in the two cases.

Table 4: Experimental results on the PHOENIX14T dataset
using the re-encode method and the original method.

| Data           | Model       | k=1   | k=3   | k=7   |
|----------------|-------------|-------|-------|-------|
| additional data | SimulSLT(-Re-encode) | 11.94 | 12.00 | 12.71 |
| additional data | SimulSLT    | 12.66 | 12.83 | 14.10 |
| without gloss  | SimulSLT(-Re-encode) | 10.23 | 10.61 | 11.73 |
| without gloss  | SimulSLT    | 11.24 | 11.37 | 12.27 |

4.6.4 The Effectiveness of Re-encode. We further explored the
impact of the re-encode method on the performance of the model.
In the first step, we conducted experiments to analyze whether the
re-encode method shown in Figure 3(b) can improve the translation
accuracy of the model compared to the original method shown in
Figure 3(a). The experimental results are shown in Table 3 (Row 5
vs Row 2) and Table 4. It can be seen that in all cases, compared
with the original method, the re-encode method can help the model
improve the translation accuracy.

Table 5: Ablation study of different fusion methods.

| Type           | with gloss (k=7) | additional data (k=7) | without gloss (k=7) |
|----------------|------------------|-----------------------|---------------------|
| Re-encode-only | 22.98            | 13.89                 | 12.01               |
| concat         | 22.89            | 13.75                 | 11.89               |
| add            | 23.14            | 14.10                 | 12.27               |

4.6.5 The impact of different fusion methods. We further explored
the impact of different fusion methods as shown in Figure 2(b) on
the performance of the model. The experimental results are shown
in Table 5. It can be found that the best translation accuracy rate is
obtained by using the direct addition fusion method.

4.7 Qualitative Results
In this section we show some qualitative results of our model. As
shown in the first and second row in Figure 6, our model performs
well when translating short sentences, but there is still some differ-
ence with the ground truth when translating long sentences (show
in Row 3 and Row 4). The front part of the long sentence predicted
by the model is quite different from the true value. As mentioned in
[5], the order of sign language video and spoken language is not the
same, and the sign language video corresponding to the former part
of spoken language semantics may be in the latter part. Therefore,
on the one hand, the model does not get the corresponding semantic
information when translating the front part of the long sentence;
on the other hand, long sentences pose a greater challenge to the
model’s context prediction ability. These two reasons lead to this
problem. In the case of lenient latency requirements, we can use
larger k to alleviate this problem. In addition, we can also alleviate
this problem by building larger datasets and models with stronger
contextual predictive capabilities.

Figure 6: Comparison of the spoken language generated by
the SimulSLT model and the reference text.

5 CONCLUSION
In this work, we developed SimulSLT, an end-to-end simultaneous
sign language translation system, which can continuously translate
sign language videos into spoken language. To achieve controllable
low-latency translation, we introduce a wait-k strategy to control the
read and write of the model. We divide the continuous sign
language video into discrete video segments by introducing an a
kind of technology based boundary predictor, which is inspired by
the way human neurons work in biology. To bridge the accuracy
gap between the simultaneous sign language translation model and
non-simultaneous sign language translation, we propose a novel
re-encode method to help the model obtain more abundant con-
textual information, which allows the video features to interact
more fully. Knowledge distillation is also used to transfer knowl-
edge from the non-simultaneous sign language translation teacher
model to the simultaneous sign language translation student model,
which further improves the performance of the model and reduces
the optimization difficulty of the model. In order to better help the
boundary predictor learn alignment information and enhance the
feature extraction capability of the encoder, we propose a series of
methods including auxiliary gloss decoding task and CTC auxil-
iary decoding. Experiments on the PHOENIX14T dataset show that
SimulSLT can achieve a translation accuracy that exceeds the lat-
est end-to-end non-simultaneous sign language translation model
while maintaining a much lower latency.

ACKNOWLEDGMENTS
This work was supported in part by the National Key R&D Program
of China (Grant No. 2018AA0100603), National Natural Science
Foundation of China under Grant No.61836002, No.62072397 and
Zhejiang Natural Science Foundation under Grant LR19F02006.
This work was also partially supported by the Huawei Noah’s Ark
Lab.
