Sign Language Recognition

Dan Guo, Shengeng Tang, Richang Hong, and Meng Wang

Abstract This chapter covers several research works on sign language recognition (SLR), including isolated word recognition and continuous sentence translation. To solve isolated SLR, an Adaptive-HMM (hidden Markov model) framework (Guo et al., TOMCCAP 14(1):1–18, 2017) is proposed. The method explores the intrinsic properties and complementary relationship among different modalities. Continuous sentence sign translation (SLT) suffers from sequential variations of visual representations without any word alignment clue. To exploit spatiotemporal clues for identifying signs, a hierarchical recurrent neural network (RNN) is adopted to encode visual contents at different visual granularities (Guo et al., AAAI, pp 6845–6852, 2018; Guo et al., ACM TIP 29:1575–1590, 2020). In the encoding stage, key segments in the temporal stream are adaptively captured. Not only RNNs are used for sequential learning; convolutional neural networks (CNNs) can be used (Wang et al., ACM MM, pp 1483–1491, 2018). The proposed DenseTCN model encodes temporal cues of continuous gestures by using CNN operations (Guo et al., IJCAI, pp 744–750, 2019). As SLT is a weakly supervised task, due to the gesture variation without word alignment annotation, the pseudo-supervised learning mechanism contributes to solving the word alignment issue (Guo et al., IJCAI, pp 751–757, 2019).

Keywords Sign language recognition · Sign language translation · Sequence-to-sequence learning · Connectionist temporal decoding · Pseudo-supervised optimization

D. Guo (✉) · S. Tang · R. Hong · M. Wang
Hefei University of Technology, University of Hefei, Hefei, China
e-mail: guodan@hfut.edu.cn; tsg1995@mail.hfut.edu.cn

T. McDaniel, X. Liu (eds.), *Multimedia for Accessible Human Computer Interfaces*,
https://doi.org/10.1007/978-3-030-70716-3_2

1 Online Early-Late Fusion Based on Adaptive HMM for Sign Language Recognition

1.1 Introduction

For sign language recognition (SLR) based on multi-modal data, a sign word can be represented by various features with existing complementary relationships among them. To investigate these complementary relationships, we present an online early-late fusion model based on an adaptive hidden Markov model (HMM) [8]. Inherent latent patterns of signs are not only associated to key gestures and body poses, but also related to the relationships among them. The proposed adaptive-HMM is designed to acquire the hidden state number of each sign through affinity propagation clustering. For complementary learning, we suggest an online early-late fusion scheme. The early fusion (feature fusion) is used to exploit the joint feature learning to achieve higher complementary scores while the late fusion (score fusion) uncovers and aggregates various scores in a weighting paradigm. This fusion is query-adaptive. Experiments verify viability on signer-independent SLR tasks with a large vocabulary. The proposed adaptive-HMM demonstrates consistent robustness in terms of performance on different dataset sizes and SLR models.

In this section, the remarkable GMM (Gaussian mixture model)-HMM model is chosen as a basic framework to tackle the isolated SLR problem. Given N signs' training data, each sign n has its own HMM $\lambda_n (1 \leq n \leq N)$, thus we have N signs' HMMs: $\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$. We use the public toolkit[1] to learn $\{\lambda_n\}$. The recognition process is implemented using the Viterbi algorithm, and the most likely sign class λ^* of observation sequence O is found by Eq. 1.

$$\lambda^* = \underset{\{\lambda_1, \lambda_2, \ldots, \lambda_n, \ldots, \lambda_N\}}{argmax} P(\lambda_n | O). \tag{1}$$

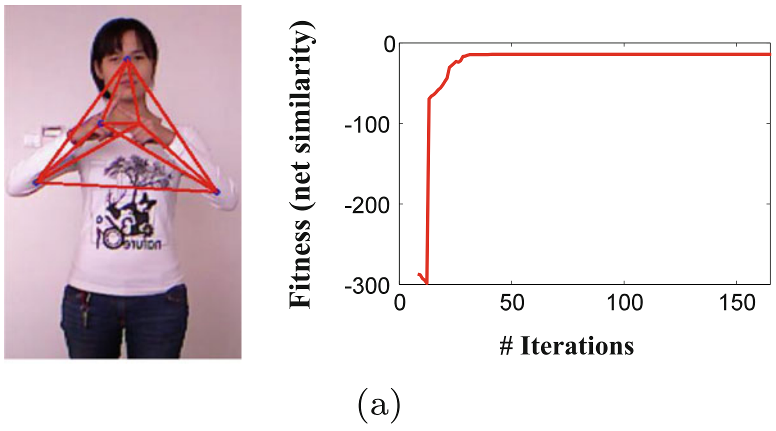
where $P(\lambda_n|O)$, learned by the model $\lambda_n (n = 1, \ldots, N)$, indicates the relevance probability of query O related to the n-th sign. To be specific, HMM-states adaptation and early-late fusion are elaborated in Sects. 1.2 and 1.3.

1.2 Adaptive HMMs

The HMM model is sensitive to its inherent latent states. To obtain a superior sign recognizer, we attempt to learn appropriate latent states for each sign word. Here, we propose HMM-state adaptation to decide the individual state number $Q_n (1 \leq n \leq N)$ for each sign model. Before learning the HMM λ_n, we split all

[1]HMM package: http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm/html. Parameters Q and M are discussed, whereas A, B and π can be handled by this code package.

the data samples of sign n into reasonable clusters and ensure an appropriate sort number for gesture variation. We adopt affinity propagation (AP) clustering [6] to adaptively acquire the centroid number of the training data. For sign n, we embed the distance measurement in the AP approach to build a frame-similarity net. The net is used to calculate the mutual responsibility and accessibility log-probability ratios between any two frames f_i and f_j. We explore the similarity function in AP to iteratively locate the best frame as exemplar f_k, which has larger responsibility weight than all other frames, until no more new exemplars appear. Thus, these best exemplars $\{f_k\}$ are taken as the centroid and we can naturally obtain the centroid number k_n (Fig. 1).



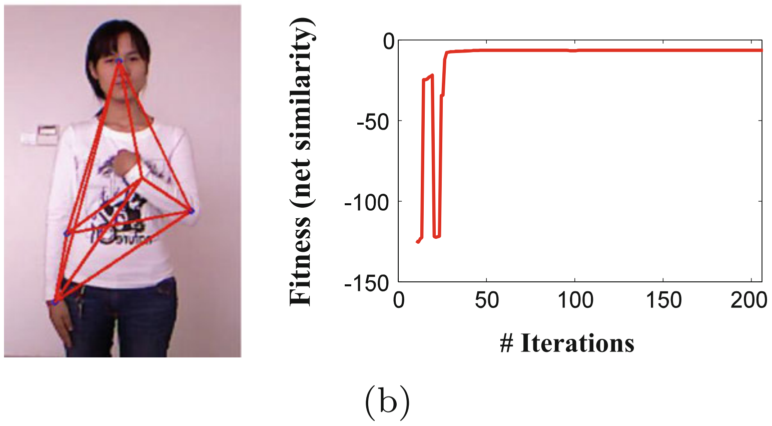


Fig. 1 Cluster convergence on SP (skeleton pair) feature with similarity computation, where "fitness" is a metric [6]. While fitness is closer to 0, clustering convergence is vastly improved. (**a**) Sign "*people*". (**b**) Sign "*I*"

Algorithm 1 Early-late fusion based on adaptive GMM−HMM (adaptive-HMM)

```
Require: $N$ signs' training sample sets; Query $O$
Ensure: the sign class of query $O$
    Training:
 1: for sign $n(1 \leq n \leq N)$ under feature $i(1 \leq i \leq m)$ do
 2:     Extract feature set $F_n^{(i)}$ from sign $n$'s training set $Set\, O_n$;
 3:     Compute the number of clusters $k_n^{(i)}$ on $F_n^{(i)}$ by AP clustering;[2]
 4:     $Q_n^{(i)} = k_n^{(i)}/M$;
 5:     Learn the GMM-HMM model $\lambda_n^{(i)} = (A, B, \pi)$ with $Set\, O_n$ and $Q_n^{(i)}$;
 6: end for
    Testing:
 7: Feature selection: e.g. remove "bad" HOG feature in the work;
 8: Obtain $O$'s remaining $m'$ score lists $\{s_O^{(i)}\}$ by SLR models $\{\lambda_n^{(i)}\}$ ;
 9: Calculate the fused score list $s_O^*$ by Eq. 2∼ Eq. 5;
10: $n^* = \arg\max_{n^* \in N} s_O^*$;
```

In the Adaptive-HMM, M denotes the cluster number of data distribution in the GMM stage and Q represents the number of latent states in the HMM stage. Classical SLR methods set M as a constant value, and typically set $M = 3$; we follow this usage. We take the hidden state number $Q_n(1 \leq n \leq N)$ as a to-be-learned factor, which reflects the characterization of key gestures. With a fixed M-component in the GMM stage of the model, the state number Q_n is set to be proportional to the number of clusters k_n, where k_n demonstrates the number of centroids of the gesture sample sequence. The proposed Adaptive-HMM is shown in step $1 \sim 5$ of Algorithm 1.

1.3 *Early-Late Fusion*

The adaptive-HMM can be applied to model the hidden states of each sign under multi-modalities, e.g., RGB images and skeletal coordinate data that are discussed in this subsection. We then discuss the complementarity of different HMM models under various feature types through the exploitation of early fusion and late fusion. The early fusion is straightforwardly implemented by concatenating various features into a combined feature as elaborated in Sect. 1.4.1. To explore the score fusion, we build the score list (a score vector) of query O within each feature type (including the combined feature). For multiple feature types $F^{(i)}(i = 1, \ldots, m)$, we obtain the **score list** set of query O by N signs' adaptive HMMs $\{\lambda_1^{(i)}, \lambda_2^{(i)}, \ldots, \lambda_N^{(i)}\}$ in Eq. 2:

$$s_O^{(i)} = [P(\lambda_1^{(i)}|O), P(\lambda_2^{(i)}|O), \ldots, P(\lambda_N^{(i)}|O)] \tag{2}$$

[2]As in [6], here we set similarity preference to media similarity in AP.

where $P(\lambda_n^{(i)}|O)$ represents the relevance of query O related to the n-th sign within feature $F^{(i)}$, $(n = 1, \ldots, N)$; it is derived from the Viterbi algorithm on $\lambda_n^{(i)}$. Thus, we obtain score lists $\{s_O^{(1)}, s_O^{(2)}, \ldots, s_O^{(m)}\}$ of query O under m feature types.

1.3.1 Feature Selection

Given that a “bad” feature can pull down the overall fusion performance, we propose a feature selection paradigm; namely, if the performance of a combined feature is superior to its single component, we abandon the “bad” component feature whose performance is worse than the combined feature. Therefore, the complementary relationship in the combined feature is maintained while filtering the redundant, “bad” information. We take the average variance of score lists on the training data as the feature selection criterion. Given a score list (a score vector), its variance means the deviation degree from its own mean value. A smaller variance means that different signs have similar scores in the list, and thus, cannot be distinguished. In contrast, a larger variance represents good discrimination power. As illustrated in Fig. 2, we build a LOO (leave-one-out) cross-validation experiment on a partial small size dataset: the 50-sign CSL dataset. Under various features, variances and average variances of score lists of a total of 1000 training samples are separately illustrated in Fig. 2a and b. The performance of variance on the feature HOG of RGB images is not as good compared to the combined feature SP (skeletal pair coordinate)-HOG. Thus, we eliminate HOG, and select SP and SP-HOG as to-be-fused features.

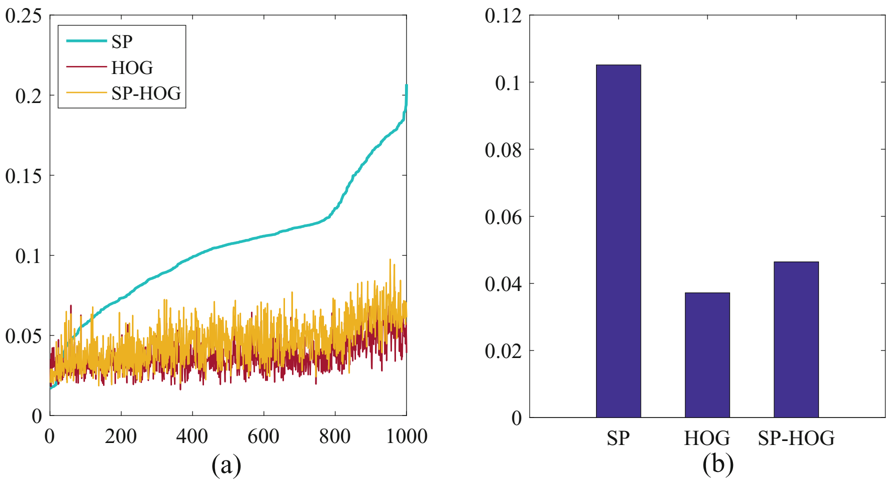


Fig. 2 Variance curves and average variances of score lists of the training samples under different features. Feature SP is superior, followed by SP-HOG and HOG. (**a**) Variance comparison with an arranged sample order sorted on the SP feature. (**b**) Average variance

1.3.2 Query-Adaptive Weighting

After feature selection, we weight the remaining m' score lists $\{s_O^{(i)}\}$. The fusion weight is inversely relative to the area of normalized sorted score curve. This is because a better $s_O^{(i)}$ is assigned a larger weight, while having a higher score on the right word label, and a much lower score on other irrelevant labels. In other words, if the arranged score list has a much sharper curve, the score list with its feature is much more discriminative and helpful. To be more explicit, we sort $s_O^{(i)}$ in decreasing order and apply min-max normalization. We denote it as $s_O^{'(i)}$ and weight on $s_O^{'(i)} (1 \leq i \leq m')$ as follows:

$$\begin{cases} s_O^{'(i)} = \dfrac{s_O^{'(i)} - mins_O^{'(i)}}{maxs_O^{'(i)} - mins_O^{'(i)}} \\ w_O^{(i)} = \dfrac{1/A_{s_O^{'(i)}}}{\sum\limits_{1 \leq i \leq n} 1/A_{s_O^{'(i)}}} \end{cases} \tag{3}$$

where $A_{s_O^{'(i)}}$ denotes the curve area of the i-th score list $s_O^{'(i)}$ under feature $F^{(i)} (1 \leq i \leq m')$. It represents that the weighting paradigm is conditioned on $s_O^{'(i)}$, i.e., the query Q itself. The weighting stage is query-adaptive and unsupervised.

1.3.3 Score Fusion

We then fuse m' score lists. The product rule typically results in better performance than other rules in biometric multi-modality fusion [15, 35]. The fusion formula is shown in Eq. 4 and a deformation in Eq. 5. Using the publicly available MATLAB package in footnote 1, we directly implement Eq. 5 in a sum format.

$$s_O^* = \left[\prod_{i=1}^{n} (s_O^{'(i)})^{w_O^{(i)}} \right], \; s.t. \sum_{i=1}^{n} w_O^{(i)} = 1 \tag{4}$$

$$s_O^* = \left[\sum_{i=1}^{n} w_O^{(i)} \cdot \log(s_O^{'(i)}) \right], \; s.t. \sum_{i=1}^{n} w_O^{(i)} = 1 \tag{5}$$

The predicted sign class of query O corresponds to the maximum value in s_O^*.

$$n^* = \underset{n^* \in N}{\arg max}\, s_O^* = \underset{n^* \in N}{\arg max} [s_{O,n}^*] \tag{6}$$

Table 1 The details of 370-sign CSL dataset

Signs	Dataset	Signer number	Repetition time	Sample number
370	Training	4	5	20×370
	Testing	1	5	5×370

where s_O^* denotes an N-dim classification vector. Its n^*-th component $s_{O,n}^*$ represents the probability of query O related to the n^*-th sign under feature $F^{(i)}$.

1.4 Experiments

1.4.1 Experiments Setup

Dataset We conducted experiments on the CSL (Chinese sign language) dataset, which is an RGB-D dataset collected using a Kinect sensor [30]. As shown in Table 1, the dataset consists of 370 signs performed 5 times by 5 signers including both men and women. The heights and gesture habits of signers are completely different. To guarantee the signer-independent test, we use the leave-one-out (LOO) cross-validation strategy to test SLR models in the experiments.

Feature Extraction This RGB-D SLR dataset contains color (RGB images) and depth (skeletal coordinates) modalities. We aim to learn the complementarity of these two modalities. In this work, we take the skeleton pair feature (D: 10-dimensional SP feature), hand feature (RGB: 51-dimensional HOG feature by PCA dimensionality reduction) and SP-HOG (RGB-D: 61-dimensional fused feature) as the basic features for each sign word.

- **Hand-crafted feature (RGB images)**: The HOG feature F_{HOG} is derived from the area of the two hands in the images by using an adaptive skin model and depth constraint as in [30].[3] Due to the high dimensionality of the original HOG feature, Principal Component Analysis (PCA) is adopted. We hold about 80% of the information energy of the dataset through PCA and obtain the 51-dimensional HOG feature. The PCA transformation matrix is acquired on the training data, and applied to the test samples.
- **Skeleton pair feature (Depth Data)**: For depth data, we first extract mutual distances of five skeleton points (head, left elbow, right elbow, left hand, and right hand), then convert them to a 10-dimensional SP distance feature F_{SP} [29]. Every signer has a different body shape. In order to unify the gesture postures of different signers, we normalize each SP vector by its maximum value. We then

[3]In this work, HOG features were extracted through OpenCV with basic parameters [30] and further optimized, e.g., some invalid frames are deleted.

obtain the SP feature sequence F_{SP} of each sign video sample. The characteristic of SP features is invariant to rotation, scaling, and translation.

- **Combined feature (RGB-D Data)**: We concatenate the F_{HOG} and F_{SP} features to create the SP-HOG feature (61-dimensional combined feature). The SP-HOG feature is considered to be early (feature) fusion.

Data Augmentation on Feature SP In the following experiments, we enhance our restricted training data with data augmentation [3]. To avoid overfitting, we investigate random Gaussian perturbation on the skeleton coordinates to enrich additional gesture positions. Given the 3D-depth skeleton point (x, y, z) extracted by the Kinect sensor, we take the coordinate x as an example to clarify Gaussian disturbance, and both y and z coordinates are tackled similarly. First, we calculate the range of x in all training samples within each sign n: $[x^n_{max}, x^n_{min}]$. Let $\Delta x^n = x^n_{max} - x^n_{min}$. Then, we set a Gaussian random variable $X \sim N(0, (\eta \Delta x^n)^2)$, where η is the disturbance parameter. We set the empirical parameter to $\eta = 0.01$. Under sign n, an additional combination of (x', y', z') coordinates of the skeleton point is generated, as shown below. By implementing data augmentation once, the original dataset is expanded to twice the original size.

$$\begin{cases} x' = & x + N(0, (\eta \Delta x^n)^2) \\ y' = & y + N(0, (\eta \Delta y^n)^2) \\ z' = & z + N(0, (\eta \Delta z^n)^2) \end{cases} \tag{7}$$

Compared Approaches We compare the proposed approach with other SLR methods, e.g., DTW [2, 25], GMM-HMM (HMMs) and Light-HMM [30]. In addition, we also compare the early-later fusion strategy with other fusion approaches, for example, early fusion for SLR [30] and late fusion [35].

- GMM-HMM [30]: For the classical GMM-HMM, a better parameter setting is $Q = M = 3$, where Q denotes the number of states in the HMM and M denotes the number of mixture models in the GMM.
- Light-HMM [30]: In order to trade off accuracy and run time, the Light-HMM determines key frames and chooses Q adaptively. Here $M = 3$ and Q is adaptive. In order to acquire excellent performance, we set Light-HMM's threshold ε_0 to 0.001 and threshold λ to the average value of the RSS score curve of parameter ε.
- DTW [2, 25]: Different from HMM calculating the probability score of query O under each sign class, DTW searches its nearest neighbor among the entire training samples and regards the sign class of the nearest neighbor as its class. In the late fusion testing of this work, the DTW score list is set as the reciprocal of the distances from all training samples.

1.4.2 Experiment with HMM-States Adaptation

We evaluate the Adaptive-HMM. The adaptation is set to HMM(Q) with adaptive Q and $M = 3$. We also evaluate adaptation HMM(M), in which M is adaptive and $Q = 3$. As shown in Table 2, compared with various HMMs, DTW is significantly more time-consuming as it compares all training samples to the to-be-identified sample, while the HMM merely learns the hidden states. Light-HMM performs worse than Adaptive-HMM due to dropping a few key frames. In the proposed adaptive HMM, the parameter Q has considerably more influence than M. Q indicates status changes, while M simulates data distribution. Due to rare samples and chaos characteristic of Gaussian simulation, the impact of M is not exceptionally clear on the CSL SLR dataset. Thus, we adopt the Adaptive-HMM(Q) as our adaptation paradigm.

1.4.3 Comparison on Different Fusion Steps

Table 3 lists different fusion strategies. In Table 4, under Recall@R=1, the precision with the SP feature is 34.82% and HOG feature just reaches 21.52%. The performance of Fusion I (early fusion) is lower than that of a single SP feature; HOG features have a negative influence. Both Fusion II, Fusion III and our early-late fusion acquire obvious improvements. These fusions, involving late (score) fusion, have learned the positive effect of complementary features. As shown in Fig. 3, the performance of Fusion II and Fusion III are close, but our fusion performance is superior. Compared with Fusion II and Fusion III, our fusion further reduces the negative impact of "bad" single feature HOG.

Table 2 Performance comparison with the single SP feature

Methods	Top 1	Top 5	Top 10	Testing time (ms/sign)
DTW	0.3159	0.5284	0.6245	1730
GMM-HMM	0.2751	0.5384	0.6583	159
LightHMM	0.2196	0.4529	0.6322	128
Adaptive-HMM (M)	0.2810	0.5404	0.6662	82
Adaptive-HMM (Q)	**0.3482**	**0.6129**	**0.7080**	88

Table 3 Setting details of different fusion strategies

Fusion I	Fusion II	Fusion III	Early-late fusion
Early fusion	Late fusion	Early-late fusion	Feature selection + Early-late fusion
SP-HOG	SP $\otimes$ HOG	SP $\otimes$ HOG $\otimes$ SP-HOG	SP $\otimes$ SP-HOG
Feature fusion	Score fusion	(feature + score) fusion	(feature + score) fusion

Table 4 Performance comparison with different fusion types

Feature	Recall@R		
	$R = 1$	$R = 3$	$R = 5$
SP	0.3482	0.5326	0.6129
HOG	0.2152	0.3403	0.3989
Fusion I [30]	0.3202	0.4571	0.5192
Fusion II [35]	0.4121	0.5627	0.6275
Fusion III	0.4140	0.5674	0.6299
Early-late fusion	**0.4532**	**0.6075**	**0.6734**

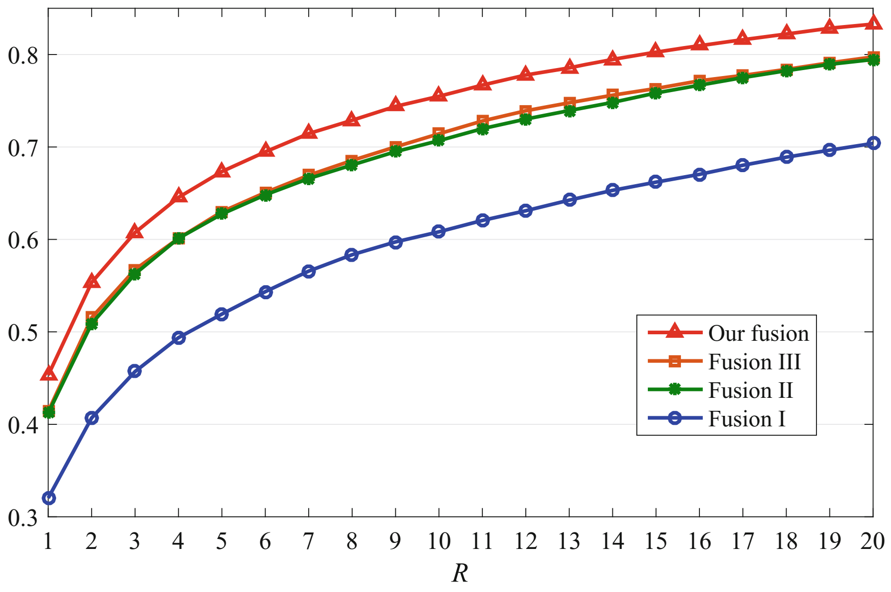


Fig. 3 Recall@R on the 370-sign CSL dataset

1.4.4 Comparison on Different Dataset Sizes

In this section, we test the Adaptive-HMM with different dataset sizes. We sample several of 370 sign words as subsets (e.g., the top 50, 100, and 200 words). As illustrated in Fig. 4a, with the increase of Recall@R, the precision improves; when the number of sign word increases, the performance drops. The differences between our fusion and other fusions are shown in Fig. 4b. Our fusion has obviously better precision and stability.

1.4.5 Comparison on Different SLR Models

As shown in Tables 5 and 6, the HOG feature achieves poor performance, and DTW still performs with the worst precision. Due to HOG delivering poor performance,

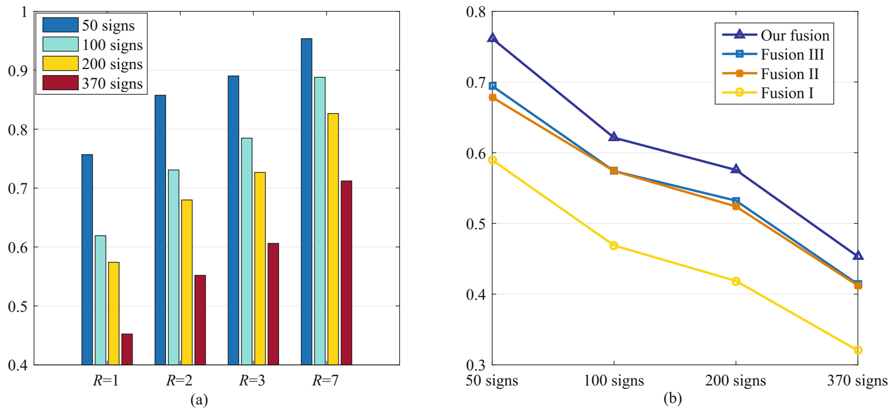


Fig. 4 (**a**) Precision comparison of our fusion with different dataset sizes. (**b**) Precision differences of our fusion to Fusions I, II and III at Recall@R=1

the performance of Fusion I (early fusion) on the 50-sign dataset is very poor. What is interesting is that Fusion II (late fusion) effectively utilizes HOG. Our fusion further improves the accuracy. To summarize, our method provides the best performance by exploiting the advantages of early and late fusion as well as complementarity. The time costs of different SLR models are shown in Table 7. Compared with other HMMs, DTW takes much more time. The time costs of various HMMs are close to each other. GMM-HMM has a stable time cost when Q is fixed with 3. Under a few key frames, the time cost of Light-HMM is higher than GMM-HMM, as its average adaptive Q is nearly $4 \sim 5$ times of GMM-HMM's Q. Compared with GMM-HMM, it has a higher computational complexity of adaptive state transitions, so the Adaptive-HMM also has a variable Q. In any case, the score fusion time is insignificant compared with the query time under different SLR models. Fusion computation is effective for online fusion.

2 Hierarchical LSTM for Sign Language Translation

2.1 Introduction

Sign Language sentence Translation (SLT) is challenging due to the specific linguistics under continuously changing gestures. To address this issue, a hierarchical LSTM (HLSTM) based encoder-decoder model is proposed [12]. It tackles different visual clues of frames, clips, and sub-sign units. As shown in Fig. 5, firstly, a 3D CNN is used to learn the temporal and spatial clues of video clips, and then the online adaptive variable length key clip mining method is used to pack sub-sign units. Afterward, we realize a temporal attention mechanism to balance the

Table 5 Fusion comparison on different SLR models with 50 signs

	DTW		GMM-HMM		Light-HMM [30]		Our adaptive HMM	
	Recall@1	Recall@4	Recall@1	Recall@4	Recall@1	Recall@4	Recall@1	Recall@4
SP	61.84%	82.08%	61.60%	88.96%	55.44%	82.00%	71.68%	93.60%
HOG	20.24%	39.68%	44.64%	68.64%	29.12%	55.92%	46.88%	68.24%
Fusion I [30]	22.88%	43.44%	58.40%	81.36%	50.40%	74.64%	58.96%	79.20%
Fusion II [35]	63.12%	82.96%	66.08%	84.96%	49.92%	75.76%	**67.84**%	**87.12**%
Fusion III	63.96%	83.76%	66.16%	86.88%	57.68%	80.08%	**69.44**%	**87.36**%
Our fusion	63.04%	83.04%	72.64%	90.40%	61.92%	83.52%	**76.16**%	**91.84**%

Table 6 Fusion comparison on different SLR models with 370 signs

	GMM-HMM			Light-HMM			Our adaptive HMM		
	Recall@1	Recall@3	Recall@5	Recall@1	Recall@3	Recall@5	Recall@1	Recall@3	Recall@5
SP	27.51%	45.56%	53.84%	21.96%	37.41%	45.29%	**34.82%**	**53.26%**	**61.29%**
HOG	21.38%	33.51%	39.61%	10.50%	19.29%	24.93%	21.52%	34.03%	39.89%
Fusion I [30]	32.00%	46.13%	52.63%	24.50%	38.49%	45.28%	32.02%	45.71%	51.92%
Fusion II [35]	37.46%	52.95%	59.39%	22.22%	36.23%	43.57%	**41.21%**	**56.27%**	**62.75%**
Fusion III	38.93%	54.42%	60.79%	29.04%	44.41%	51.72%	**41.40%**	**56.75%**	**62.99%**
Our fusion	**41.50%**	**57.36%**	**63.91%**	**33.88%**	**49.64%**	**56.67%**	**45.32%**	**60.75%**	**67.34%**

Table 7 Time comparison on 50 signs. Fusion time in this table merely indicates time of fusion computation

Avg. testing time (s)	DTW	GMM-HMM	Light-HMM	Our adaptive HMM
SP	1.730	0.088	0.128	0.159
HOG	8.495	0.123	0.399	0.179
Fusion I	9.025	0.124	0.217	0.156
Fusion II	**0.014**	**0.011**	**0.011**	**0.011**
Fusion III	**0.015**	**0.011**	**0.011**	**0.011**
Our fusion	**0.014**	**0.011**	**0.011**	**0.011**

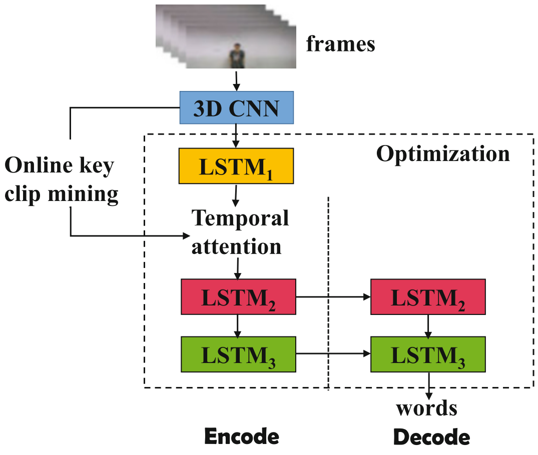


Fig. 5 The overall framework of HLSTM

relationship among sub-sign locations. Finally, two LSTM layers are used to recurse sub-sign semantics. After condensing the original visual features extracted by the 3D CNN and top LSTM layer, the bottom two LSTM layers have less computational complexity.

2.2 *Online Key Clip Mining*

Discriminative motion patterns sparsely occur throughout video, such as sign speed, habits of signers, and special characteristics of sign words. In this work, we attempt to automatically obtain the variable-length key clips, rather than fixed interval for key frames or volumes segmentation [31, 36]. We employ a low-rank approximation method [30] to calculate the linear correlation of the frame sequence, which is implemented by calculating the residual sum of square (RSS) of feature ε between the previous frame and the current frame.

Given feature sequence $F = [f_1, f_2, \cdots, f_n]$ of a video, we model a correlation matrix M to compute the residual error ε_c of feature f_c at time step c. Based on the subset $F_c = [f_1, f_2, \cdots, f_c]$, ε_c and M are initialized as $\varepsilon_1 = 0$

and $M = (\boldsymbol{f}_1^T \boldsymbol{f}_1)^{-1}$. The correlation coefficient $\boldsymbol{\pi}_c$ and residual error ε_c at time step c ($2 \le c \le n$) are learned by:

$$\begin{cases} \boldsymbol{\pi}_c = MF_{c-1}{}^T \boldsymbol{f}_c \\ \varepsilon_c = (\boldsymbol{f}_c - F_{c-1}\boldsymbol{\pi}_c)^T (\boldsymbol{f}_c - F_{c-1}\boldsymbol{\pi}_c) = \left\| \boldsymbol{f}_c - F_{c-1} M F_{c-1}{}^T \boldsymbol{f}_c \right\|^2 \end{cases} \tag{8}$$

Then, the matrix M is updated as follows:

$$M = \begin{bmatrix} M + \boldsymbol{\pi}_c^T \boldsymbol{\pi}_c \varepsilon_c & -\boldsymbol{\pi}_c / \varepsilon_c \\ -\boldsymbol{\pi}_c^T \varepsilon_c & 1/\varepsilon_c \end{bmatrix} \tag{9}$$

where M indicates the inherent linear correlation of F_c, $\boldsymbol{\pi}_c$ builds the relevance of F_{c-1} to $\boldsymbol{f}_c$, and $F_{c-1}\boldsymbol{\pi}_c$ is the approximate reconstruction of $\boldsymbol{f}_c$ by using F_{c-1} at time step c. Thus, we acquire $\varepsilon = [\varepsilon_1, \cdots, \varepsilon_c, \cdots, \varepsilon_n]$.

Previous works captured optimal subsets by selecting discrete frames; however, as shown in Fig. 6, the RSS curve tackles the continuous key and non-key variable-length clips (e.g., invalid segments for word-to-word conversion). Each peak on the RSS variable curve represents the local maximum gain of continuous variations. We keep the monotonously increasing part of ε curve as profit—**a key clip**, because the residual error increases, it cannot be replaced by the previous frame. Thereafter, the ε gradually decreases along the monotonically decreasing portion of the curve, where the monotonically increasing parts are denoted as **non-key clips**. This indicates that the reduced portion can be linearly reconstructed from the

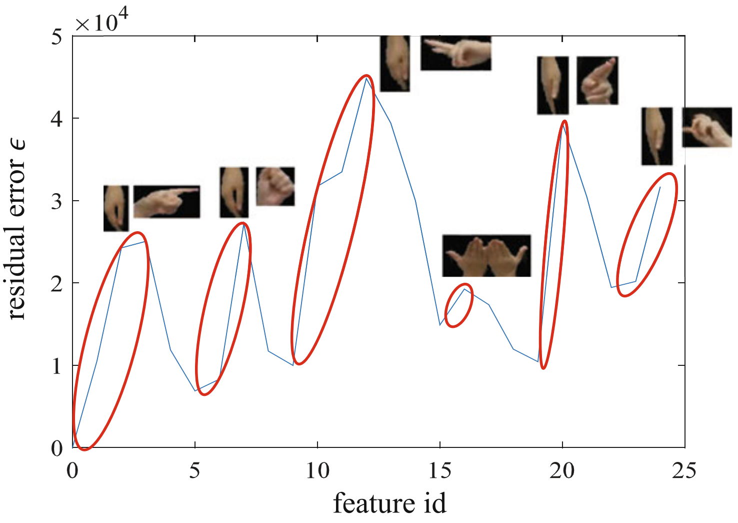


Fig. 6 Curve of ε of a video. Each peak corresponds to a sign's discriminative gesture

previous frame's downward error. To summarize, we avoid over-learning of non-key fragments in videos.

2.3 Hierarchical LSTM Encoder

The HLSTM model is a three-tier LSTM encoder architecture. The top $LSTM_1$ is responsible for extracting the recurrent representation based on the 3D spatio-temporal features $F = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n]$ obtained by the well-known C3D [27]. And then, with the use of pooling and attention-based weighting, we condense the length of $LSTM_1$ features into n'' for $LSTM_2$ and $LSTM_3$. Finally, $LSTM_2$ is mainly used for visual embedding during the encoding stage, and $LSTM_3$ is used for modeling word embedding during the decoding stage and for sequence learning. $LSTM_2$ and $LSTM_3$ are both used in the encoding and decoding phases. Their parameters are shared in these two phases.

2.3.1 Hierarchical Encoder

The input video frames $(f_1, \cdots, f_N)$ are encoded by using both CNN and LSTM modules. A visual embedded representation V is learned by:

$$\begin{aligned} V &= \theta_{lstms}[\mathcal{G}(\theta_{cnn}(f_1, \cdots, f_N))] \\ &= \theta_{lstms}[\mathcal{G}(\boldsymbol{f}_1, \boldsymbol{f}_1, \cdots, \boldsymbol{f}_n)] \\ &= \theta_{LSTM_3, LSTM_2, LSTM_1}[(\boldsymbol{f'}_1, \cdots, \boldsymbol{f'}_{n'})] \\ &= \theta_{LSTM_3, LSTM_2}(\widetilde{h}_1, \cdots, \widetilde{h}_{n''}) \\ &= (v_1, \cdots, v_{n''}) \end{aligned} \tag{10}$$

where $\mathcal{G}(\cdot)$ represents the key clip mining operation. As shown in Table 8, N, n and n' are of variable length. We set $n'' = l_{ave}$, where l_{ave} is the average of features for all training videos. $\{\widetilde{h}_1, \cdots, \widetilde{h}_{n''}\}$ is the inputs for $LSTM_2$, which is described in detail in the following section about polling and attention-based weighting. $\{v_1, \cdots, v_{n''}\}$ is the hidden states of $LSTM_3$ during the encoding phase.

Table 8 Parameter description

Symbol	Description
N	The number of original frames of a video
n	The number of extracted features
n'	The number of selected features by $\mathcal{G}(\cdot)$
n''	The number of encoding time steps of $LSTM_2$

2.3.2 Pooling Strategy

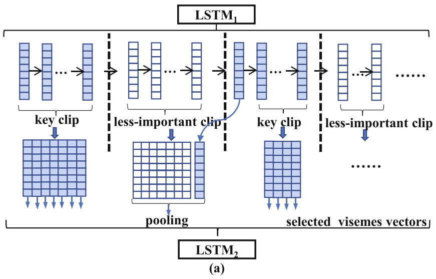
To condense less important clips, the outputs $\{h_t\}(t \in [1, n])$ of $LSTM_1$ are compressed for $LSTM_2$. As shown in Fig. 7, h_t is taken as an independent sub-sign vector if it belongs to key clips, and we perform pooling on the non-key block if h_t belongs to non-key clips. Each non-key block includes consecutive, less important clips and the first frame of the next adjacent key clip. The pooled feature block is defined as $\{h_t\}$ $(t \in [t_1^*, t_{T_c}^*])$, where T_c is the length of the pooling block. Three pooling schemes can be applied as follows:

- **Key-pooling:** The last time stamp of the block $h'_t = h_{key}$ is directly fed into $LSTM_2$. This eliminates the effect of the non-key clips.
- **Mean-pooling:** Take the average vector of the block along the time dimension as $h'_t = h_{mean}$, which balances the recurrent output of each block.
- **Max-pooling:** The maximization of the block $h'_t = h_{max}$ highlights the block's salient response.

After pooling, we fix the recurrent sequence $\{h'_t\}$ $(t \in [1, n'])$ with variable n' for different video samples into the same length n''. There are two processes: if $n' < n''$, the sequence is filled with zero-padding vectors; if $n' \geq n''$, $\{h'_t\}$ is evenly sampled to n''.

2.3.3 Attention-Based Weighting

Figure 8 describes the weighting mechanism which shows the impact of each source position on the time dimension. The learnable attention vector $W \in \mathbb{R}^{T_e}$ is modelled by end-to-end training $\widetilde{h}_t = w_t \cdot h'_t$, where h'_t is the pooled vector, and $T_e = n''$. We denote the proposed HLSTM with or without this attention module as HLST and HLSTM-attn, respectively.



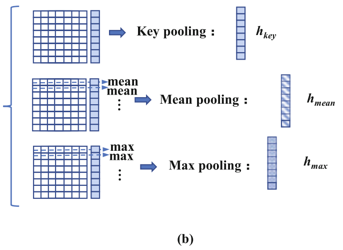


Fig. 7 Illustration of pooling strategy on non-key clips. (**a**) The pooling process. (**b**) Different pooling strategies

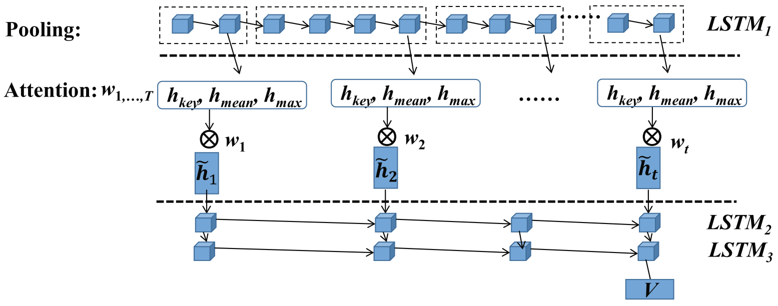


Fig. 8 Attention-based weighting mechanism

2.4 Sentence Generation

For the decoding stage, we apply $LSTM_2$ and $LSTM_3$ for sentence generation, where $LSTM_3$ recurrently outputs the generated sentences. With V obtained in the encoding stage, the decoder outputs the conditional probability of the generated sentence $(y_1, \cdots, y_m)$ as follows:

$$p(y_1, \cdots, y_m|V) = \prod_{t=1}^{m} p(y_t|v_{n''+t-1}, y_{t-1}) \tag{11}$$

We take zero-padding vectors as visual input of $LSTM_2$. $LSTM_3$ starts with $<BOS>$, and then inputs the previous word. During training, $LSTM_3$ is fed with the previous ground-truth word at each step. During the test, we select the current word (y_t) with the highest probability from the output (z_t) of $LSTM_3$ in Eq. 12, and its word embedding is the input at the next time step.

$$p(y_t|z_t) = \frac{exp(W_y z_t)}{\sum\limits_{z_t'=V} exp(W_y z_t')} \tag{12}$$

We utilize the entropy of the generated sentences to learn the model parameters θ. Loss optimization is performed during the decoding phase.

$$\theta^* = \underset{\theta}{\text{argmax}} \sum_{t=1}^{m} p(y_t|v_{n''+t-1}, y_{t-1}; \theta) \log p(y_t|v_{n''+t-1}, y_{t-1}; \theta) \tag{13}$$

Table 9 Details of the USTC-CSL dataset

		Signers	Sentences	Samples
Split I	Train	40	100	$40 \times 100 = 4000$
	Test	10	100	$10 \times 100 = 1000$
Split II	Train	50	94	$50 \times 94 = 4700$
	Test	50	6	$50 \times 6 = 300$

2.5 Experiment

2.5.1 Experiment Setup

Dataset The USTC-CSL dataset consists of sign videos that cover 100 daily sentences in Chinese sign language (CSL).[1] There are 50 signers to play each sentence, resulting in 5000 videos. And each sentence comprises $4 \sim 8$ (average 5) sign words (phrases). The vocabulary size is 179. As shown in Table 9, the dataset is split as follows: **Split I—signer independent test:** Videos played by 40 signers are taken as the training set, and videos played by the remaining 10 signers are taken as the test set. In other words, the training and test sets have the same sentences but are played by different signers. **Split II—unseen sentences test:** 6 sentences are selected in which words separately appeared in the remaining 94 sentences so that a word appears in different sentences with different occurrence orders and usages.

Evaluation Metrics **Precision** reflects the ratio of correct sentences. ***Acc-w*** calculates the mean ratio of the correct word to the reference word in a sentence. Word error rate (**WER**) [4] is used to measure the minimum number of operations to change a generated sentence to reference. Semantic evaluation metrics widely used in NLP, NMT and image captioning are also reported, such as **BLEU**, **METEOR**, **ROUGE-L** and **CIDEr**.

2.5.2 Model Validation

We set the LSTM hidden state to $n_{hid} = 1000$. Features are extracted by C3D, which crops from every 16 frames with 8 frames overlapping [27].

Evaluation on Pooling Strategies The results in Table 10 demonstrate the properties of the pooling schemes. Key-pooling maintains the recurrent character on the time dimension, mean-pooling averages the recurrent output, and max-pooling emphasizes the significant responses. For Split I, mean-pooling is superior, whereas max-pooling performs better for Split II. Thus, we set mean-pooling and max-pooling for Split I and Split II, respectively.

[1] http://mccipc.ustc.edu.cn/mediawiki/index.php/SLR_Dataset.

Table 10 Comparison on different pooling strategies

Pooling strategy	Precision on Split I	*Acc-w* on Split II
Key-pooling	0.920	0.479
Mean-pooling	**0.924**	0.458
Max-pooling	0.912	**0.482**

Table 11 Comparison on different encoder frameworks

Model	Precision
S2VT ($n = 21$)	0.897
S2VT ($n = 66$)	0.850
S2VT (3-layer, $n = 21$)	0.903
S2VT (3-layer, $n = 66$)	0.854
HLSTM (SYS sampling)	0.910
HLSTM	**0.924**
HLSTM-attn	**0.929**

Evaluation on n'' Settings In the experiments, 66 is the maximum length of video C3D features of all training samples, and 21 is the average length. Note that N, n and n' are variable length for different videos. When $n'' = 66$, it recursively leads to all the sequence representations; if $n'' = 21$, compression features provide an average length. As shown in Table 11, $n'' = 21$ performs better compared to $n'' = 66$. When $n'' = 66$, there is little benefit in adding useless padding vectors to the $LSTM_2$ and $LSTM_3$ layers.

Evaluation on Encoder Frameworks Table 11 lists five similar but different encoding methods. Observing Table 11, encoders with fixed LSTM length have lower performance, such as S2VT and S2VT (3-layer). We extend S2VT to S2VT (3-layer) and make further comparisons. The S2VT (3-layer) encoder is a 3-layer LSTM of equal length, while the 2-layer decoder as our own. The hierarchical recurrent encoder performs better. In addition, the settings of the 3-layer encoder is better than the 2-layer as it combines the recurring features by the top layer LSTM. The system sampling is inferior to variable length key clip mining, and HLSTM-attn is superior with temporal attention.

2.5.3 Comparison to Existing Methods

Summary on Seen Sentences We compare the proposed approach with the following methods: **LSTM&CTC** model,[2] **S2VT** [28], **LSTM-E** [22], **LSTM-Attention** [33] and **LSTM-global-Attention** [21]. In the following experiments, if not explicitly stated, our HLSTM used only C3D features without temporary attention. HLSTM selects the mean pooling for Split I and the max pooling for Split II. As for extensions, HLSTM (SYS Sampling) has removed the selection of

[2]https://github.com/baidu-research/warp-ctc

key clips and directly inputs the outputs of $LSTM_1$ into $LSTM_2$ through system sampling in HLSTM. HLSTM-attn introduces temporal attention into HLSTM. The HLSTM models perform better than the others. Among different settings, HLSTM-attn performs the best (Table 12).

Experiment on Unseen Sentences Unseen sentences give a new challenge for SLT than seen sentences. In the test Split II, the words appear in different videos. Various variants of HLSTM are still better than other methods at identifying more meaningful words, as shown in Table 13.

3 Dense Temporal Convolution Network for Sign Language Translation

3.1 Introduction

Due to the lack of precise mapping annotation between visual actions and text words, sign language sentence translation (SLT) is a weakly supervised task. Aligning sign actions and the corresponding words is the goal of this work; thus, Dense Temporal Convolutional Network (DenseTCN) is proposed to capture actions in a hierarchical view [11]. The overview of DenseTCN is shown in Fig. 9; a temporal convolution (TC) unit is used to learn the short-term correlation between adjacent features, and is further expanded into a dense hierarchical structure. Finally, the output sets of all the previous layers will be integrated together at the kth TC layer. The merits of DenseNet consist of: (1) the deeper TC layer captures longer-term temporal context through hierarchical content aggregation, and (2) leveraging short-term and extended long-term sequential learning is useful to address the sequential alignment in the SLT task. The CTC loss and a fusion strategy are used to refine the predicted sentences.

3.2 DenseTCN

Different from classical RNN cells, TC is a convolutional operation to tackle short-term temporal calculation. DenseNet captures long-term temporal context through hierarchical TC layers. Taking the $k-$th TC layer as an example, the technical details of TC are shown in Fig. 10; using TC on the input feature matrix $H_k = \{h_i\}_{i=1}^{M} \in \mathbb{R}^{k\times M\times d'}$, it is transformed into the output feature matrix $\{h'_i\}_{i=1}^{M} \in \mathbb{R}^{M\times d'}$, where M is the length of the feature sequence and d' is the feature dimension. Specifically, each TC layer embeds all the outputs of previous TC layers into a new compact representation. The parameters of the kth$(k > 0)$ TC layer are defined as [number of filters, k, n, the input feature dimension, $padding$, $stride$]. As shown in Fig. 10,

Table 12 Evaluation under Split I for seen sentence recognition

	Precision	CIDEr	BLEU-4	BLEU-3	BLEU-2	BLEU-1	ROUGE-L	METEOR
LSTM&CTC	0.858	8.632	0.899	0.907	0.918	0.936	0.940	0.646
S2VT	0.897	8.512	0.874	0.879	0.886	0.902	0.904	0.642
S2VT (3-layer)	0.903	8.592	0.884	0.889	0.896	0.911	0.911	0.648
HLSTM (SYS)	0.910	8.907	0.911	0.916	0.922	0.935	0.938	0.683
HLSTM	**0.924**	**9.019**	**0.922**	**0.927**	**0.932**	**0.942**	**0.944**	**0.699**
HLSTM-attn	**0.929**	**9.084**	**0.928**	**0.933**	**0.938**	**0.948**	**0.951**	**0.703**

Table 13 Evaluation under Split II for unseen sentence translation

	Acc-w	CIDEr	BLEU-3	BLEU-2	BLEU-1	ROUGE-L	METEOR	WER
LSTM&CTC	0.332	0.241	0.039	0.124	0.343	0.362	0.111	0.757
S2VT	0.457	0.479	0.135	0.258	0.466	0.461	0.189	0.670
S2VT (3-layer)	0.461	0.477	0.145	0.265	0.475	0.465	0.186	0.652
HLSTM (SYS)	0.459	0.476	0.185	0.293	0.463	0.462	0.173	**0.630**
HLSTM	**0.482**	**0.561**	**0.195**	**0.315**	**0.487**	**0.481**	**0.193**	0.662
HLSTM-attn	**0.506**	**0.605**	**0.207**	**0.330**	**0.508**	**0.503**	**0.205**	0.641

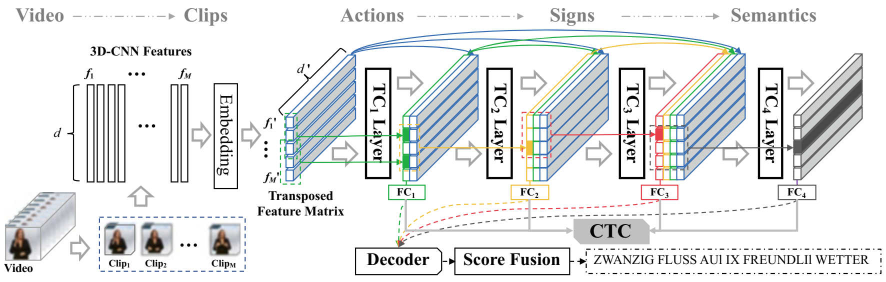


Fig. 9 The overview of DenseTCN with $K = 4$ layers for SLT

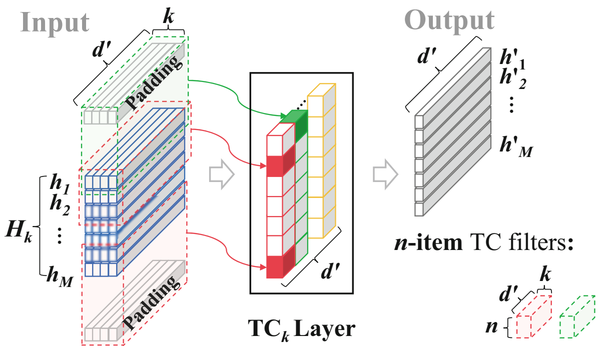


Fig. 10 Detailed operations of the TC$_k$ layer (e.g. $k = 4$, $n = 3$)

we set the parameters to $[d', k, 3, d', 1, 1]$, where $n = 3$ indicates associating three adjacent features to realize short-term awareness. H_k and O_k represent the input and output of the kth TC layer, respectively. The calculation of the TC layer is formulated as follows:

$$\begin{cases} H_0 = \mathcal{F} \in \mathbb{R}^{M\times d}, & if\ k = 0 \\ H_k = [O_{k-1}, O_{k-2}, \ldots, O_0] \in \mathbb{R}^{k\times M\times d'}, & else; \\ O_0 = \mathbf{\Phi}(H_0) \in \mathbb{R}^{M\times d'}, & if\ k = 0 \\ O_k = TC_k(H_k) \in \mathbb{R}^{M\times d'}, & else; \end{cases} \tag{14}$$

where $H_0 = \mathcal{F}$ denotes the original features and function $\mathbf{\Phi}$ denotes a FC layer that transforms the d-dimension features to d'. After each TC layer, we employ activation functions and Dropout [20] to avoid over-fitting. By cascading the outputs of all previous calculation layers, DenseNet expands the temporal receptive fields in the hierarchical design.

3.3 Sentence Learning

3.3.1 CTC Decoder

Each TC layer is followed by a fully connected (FC) layer, which converts each densely encoded feature into a word vocabulary to predict consecutive possible words. Voc is a set of all the words in the training set. and we add a blank word '_' to the vocabulary Voc to build a new vocabulary Voc'=$Voc \cup \{\text{'\_'}\}$.

$$p_k = FC_k(O_k) = O_k \cdot W_k + b_k \ , \tag{15}$$

where $p_k \in \mathbb{R}^{M \times w} = \{p_k^i\}_{i=1}^M$ is a predicted score matrix of the kth TC layer and w is the size of the vocabulary Voc'.

CTC [7] is adopted as the objective function to decode sentences. CTC applies a many-to-one mapping operation $\mathcal{B}$, which deletes the blank words and repeated words in $\boldsymbol{\pi}_k$, e.g. $\mathcal{B}(\_\ a\ a\ \_\ \_\ pencil)= \{a\ pencil\}$, to convert $\boldsymbol{\pi}_k$ into a variable sentence $\mathcal{Y} = \{apencil\}$. Therefore, the probability of a labeling $\mathcal{Y} = (y_1, y_2, \ldots, y_L)$ containing L words is the sum of the probability that all words are aligned as follows:

$$\Pr(\mathcal{Y}|p_k) = \sum_{\boldsymbol{\pi}_k \in \mathcal{B}^{-1}(\mathcal{Y})} \Pr(\boldsymbol{\pi}_k|p_k) \tag{16}$$

where $\mathcal{B}^{-1}(\mathcal{Y}) = \{\boldsymbol{\pi}_k|\mathcal{B}(\boldsymbol{\pi}_k) = \mathcal{Y}\}$ involves all possible paths $\{\boldsymbol{\pi}_k\}$. And the probability of a path $\boldsymbol{\pi}_k$ is defined as follow:

$$\Pr(\boldsymbol{\pi}_k|p_k) = \prod_{j=1}^{M} \Pr(\pi_{k,j}|p_k), \forall \pi_{k,j} \in Voc' \tag{17}$$

where $\pi_{k,j}$ is the jth element of $\boldsymbol{\pi}_k$.

A hierarchical CTC optimization is the novelty of this work. Let $P = \{p_k\}_{k=1}^K$ be the input of all the CTC decoders, where K is the depth of the DenseTCN, and the total CTC loss is defined as follows:

$$\mathcal{L}_{CTC} = -\log \Pr(\mathcal{Y}|P) = -\sum_{k=1}^{K} \log \Pr(\mathcal{Y}|p_k) \tag{18}$$

3.3.2 Score Fusion and Translation

Until now, we obtain a probability score set $P = \{p_k\}_{k=1}^K$, where $p_k \in \mathbb{R}^{M \times w}$ and w is the size of vocabulary. We choose the *softmax* operation to normalize each probability p_k, and further sum all the normalized variables of $\{p_k\}$.

$$p^i_{fusion,j} = \frac{1}{K}\sum_{k=1}^{K}\frac{e^{p^i_{k,j}}}{\sum_{j'=1}^{w} e^{p^i_{k,j'}}} \tag{19}$$

Next, we use the function *argmax* on p^i_{fusion} and output the ith word classification label with the maximum value. At last, we have to delete the blank '_' and reduplicate words by the above-mentioned two-stage greedy strategy, and output the final generated sentence.

3.4 Experiments

3.4.1 Datasets

We evaluated our method on the German sign language dataset (PHOENIX),[3] which involves daily news and weather forecasts in German sign language. As shown in Table 14, it consists of 6841 videos executed by 9 signers.

3.4.2 Evaluation Metrics

Word error rate (WER) is a widely used metric to measure the similarity of two sentences. The distance from a generated sentence to the ground-truth is required to be the minimal operations of substitution (S), deletion (D), and insertion (I). We use G to indicate the number of words in the ground truth, and WER is formulated as follows:

$$\text{WER} = (S + D + I)/G \times 100\% \tag{20}$$

When WER is lower and accuracy is higher, the model is better. Further, auxiliary evaluations *del* and *ins* represent the ratio of delete and insert operations as follows:

$$del = D/G \times 100\%, \quad ins = I/G \times 100\% \tag{21}$$

Table 14 Details of PHOENIX dataset

	Signers	Sentences	Videos	Words
TRAIN	9	5672	5672	1231
VAL	9	540	540	461
TEST	9	629	629	497

[3] https://www-i6.informatik.rwth-aachen.de/$\sim$koller/RWTH-PHOENIX/.

3.4.3 Implementation Details

For data processing, each video is divided into clips with 8 frames and an overlapping 4 frames to get 190,536 clips of the TRAIN subset, 17,908 clips of the VAL subset and 21,349 clips of the TEST subset of the PHOENIX dataset. We extract clip features with a dimension of 512 through an 18-layer 3D-ResNet [13] pre-trained on the SLR dataset [34]. The linear function $\boldsymbol{\Phi}$ transforms d-dimensional features to d'. Here, d=512. The parameters of the kth($k > 0$) TC layer are defined as [number of filters, k, n, the input feature dimension, $padding$, $stride$]. As mentioned earlier, the parameters of the kth TC are set to $[d', k, 3, d', 1, 1]$. We discuss d' in the latter Sect. 3.4.4. For the PHOENIX dataset, the input feature dimension size is 512, and the size of vocabulary is 1232. The activation function ReLU [20] is adopted after each TC layer. We train the DenseNet model with ADAM optimizer and dropout ratio $\rho = 0.5$ on each TC layer. The initialized learning rate is 10^{-4} and the weight attenuation is 10^{-5}. After 30 training epochs, the learning rate changes to be 0.9 times of the original. The training is constrained to stop while the learning rate is lower than 10^{-6}. Dropout is not considered during the testing phase.

3.4.4 Depth Discussion

In the experiments, the minimum network depth K is 1 and the maximum is 16. We test different depths in the two cases with or without (w/o) dropout. As shown in Fig. 11, the experimental results demonstrate that on the PHOENIX VAL dataset, the performance with dropout and a deeper dense network is better. Observing the experimental results, we set the empirical parameter K to 10, and embedding size d' to 512 for the PHOENIX dataset.

3.4.5 Comparison

As shown in Table 15, the symbol ▲ marks the models that introduce other features, such as “face image”; △ indicates that additional offline optimizations are used, e.g., multiple EM iterations are used for weak supervision in the hybrid CNN-HMM (CNN-Hybrid) model [19]. We analyze the differences between the models in Table 15. HMM-based models include HOG-3D and CMLLR, which employ hand-crafted features [16]. Based on deep features, Cui et al. developed a three-steps training optimization named staged-Opt [5]. 1M-Hands [17] and SubuNets [1] consider more visual clues, i.e., both hands and global images. LS-HAN solves the SLT task by an attention mechanism [14]. Through the offline EM optimization, [24] Dilated-CNN was trained five times. CTF-SLT [32] applies BGRU and TCOV to tackle long-term and short-term sequential learning respectively. Dense TC operation and the hierarchical CTC optimization are the main contributions of this work. Our proposed approach is superior to other methods. Figure 12 shows

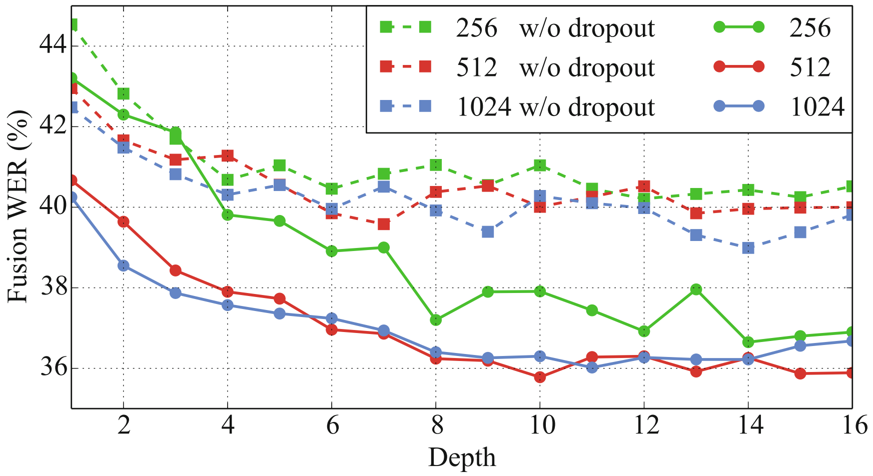


Fig. 11 Performance of DenseTCN with different depths on PHOENIX VAL set

Table 15 Evaluations under PHOENIX

	VAL		TEST	
Methods	*del/ins*	WER	*del/ins*	WER
HOG-3D ▲	25.8/4.2	60.9	23.2/4.1	58.1
CMLLR ▲	21.8/3.9	55.0	20.3/4.5	53.0
1M-Hands ▲ △	16.3/4.6	47.1	15.2/4.6	45.1
CNN-Hybrid ▲ △	12.6/5.1	38.3	11.1/5.7	38.8
Staged-Opt ▲ △	13.7/7.3	39.4	12.2/7.5	38.7
SubuNets ▲	14.6/4.0	40.8	14.3/4.0	40.7
Dilated-CNN △	8.3/4.8	38.0	7.6/4.8	37.3
LS-HAN	–	–	–	38.3
CTF-SLT	12.8/5.2	37.9	11.9/5.6	37.8
DenseNet*	–	49.7	–	49.2
Our DenseTCN	10.7/5.1	**35.9**	10.5/5.5	**36.5**

▲: Other modality, △: Extra supervision

the translation process in DenseTCN with $K = 10$ layers. With reference to the "ground-truth", the WER value of the translated sentence of each TC layer is in the range of 50% to 8%. As a result, the deletion (D) words ("MOEGLICH" and "VERSCHIEDEN"), the insertion (I) word ("UNTERSCHIED"), and the substitution (S) words ("WIND" and "loc-REGION") are modified with all the TC layers.

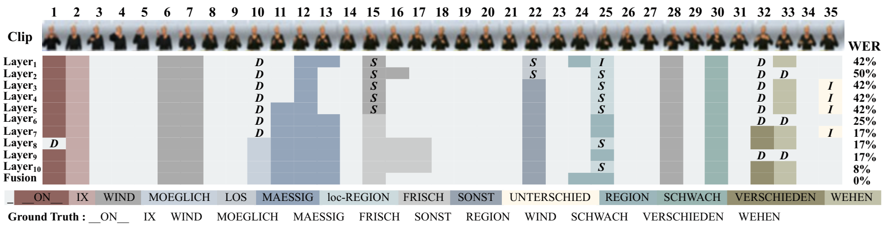


Fig. 12 The prediction results of the DenseTCN with depth $K = 10$

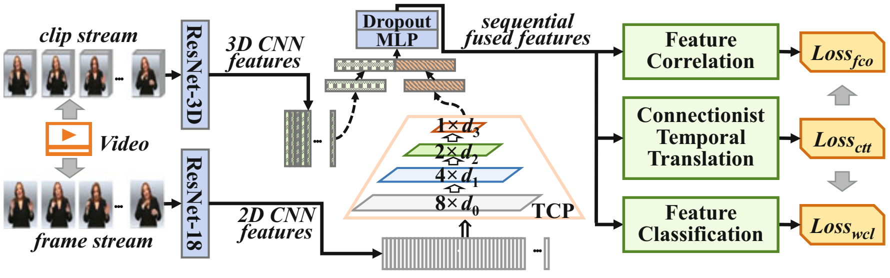


Fig. 13 The framework of the proposed CTM method for online SLT. Except for the Temporal Convolution Pyramid (TCP) module for feature extraction, the CTTR (Connectionist Temporal TRanslation) module is designed to generate sequential words that are further taken as supervision to optimize FCLS (Feature CLaSsification) and FCOR (Feature CORrelation) modules. Thus, the entire CTM method is learned in a pseudo-supervised mode

4 Joint Optimization for Translation and Sign Labeling

4.1 Introduction

Online sign translation faces challenges of hybrid semantic learning, such as visual representation, textual grammar, and sign linguistics. A Connectionist Temporal Modeling (CTM) framework is proposed for sentence translation and sign labeling [10]. As shown in Fig. 13, in order to obtain short-range temporal correlations, we designed a Temporal Convolution Pyramid (TCP) module to convert 2D CNN features to pseudo 3D′ features. After feature fusion of 2D and pseudo 3D′ features, the fused features are fed into three optimization modules in the CTM framework, i.e., Connectionist Temporal TRanslation (CTTR), Feature CLaSsification (FCLS), and Feature CORrelation (FCOR) for long-range sequential learning. Besides, dynamic programming is embedded into the decoding process, which maps sequential features to sign labels and generates sentences. During the connectionist decoding process, we regard the classification labels of clips as pseudo-labels, which are used as pseudo-supervised hints in the end-to-end framework. To be specific, we designed a joint objective function combining $\mathcal{L}_{fcor}$, $\mathcal{L}_{cttr}$, and $\mathcal{L}_{fcls}$ to optimize the decoding phase.

4.2 Clip Feature Learning in Videos

In this part, we clarify the clip-level feature extraction of two-stream CNNs, i.e., the alignment and fusion of 2D and 3D features. At first, motivated by the $\triangle t$-gram language model in the field of natural language processing, we propose a Temporal Convolution Pyramid (TCP) module to obtain contiguous features. The TCP gradually condenses temporal cues via multi-layer convolution operations with contiguous $\triangle t$-items. We use an l-layer TCP to transform the original 2D frame features $\{\boldsymbol{f_{2d}}\} \in \mathbb{R}^{d_0 \times \mathcal{N}}$ into pseudo clip-level features $\{\boldsymbol{f'_{3d}}\} \in \mathbb{R}^{d_l \times \frac{\mathcal{N}}{\prod_{i=1}^{l}(s_i)}}$. The embedding process of features in TCP is expressed as Eq. 22.

$$\{\boldsymbol{f'^{n}_{3d}}\}_{n=1}^{N} = TCL_{\Phi_l}\left\{ \cdots \left[TCL_{\Phi_1}(\boldsymbol{f^{n}_{2d}}|_{n=1}^{\mathcal{N}}) \right] \right\} \tag{22}$$

where Φ_i represents the parameter of TCL_i in TCP, and Φ_i =($ch_i, d_i, \triangle t_i, s_i, pad_i$) denotes the format of a convolutional parameter (number of channels, height, width, stride, and padding). $d_i \times \triangle t_i$ denotes the convolutional kernel size, and s_i is the temporal sliding window. Here, $d_{i+1} = ch_i$, i.e., the kernel size for the (i+1)-th layer, is set to the output dimension of the i-th layer. In other words, TCP is a combination of 2D spatial and 1D temporal convolution to compact a 2D clip into a 3D feature. We deem this operation as pseudo 3D′ features, rather than real 3D CNN operations.

As shown in Fig. 14, a three-layer TCP is realized with ($\triangle t_i = 2$)-gram and as non-overlapping ($s_i = 2$); the frame-level 2D features are transformed into a short clip-level 3D feature. After that, we fuse the $\boldsymbol{f'}_{3d}$ and $\boldsymbol{f}_{3d}$ by the MLP, which can be formulated as Eq. 23.

$$\boldsymbol{F}_{fus} = \{\boldsymbol{f}_n\}_{n=1}^{N} = MLP(\boldsymbol{f}'_{3d} \oplus \boldsymbol{f}_{3d}) \tag{23}$$

where $\oplus$ is the concatenation operation for feature vectors.

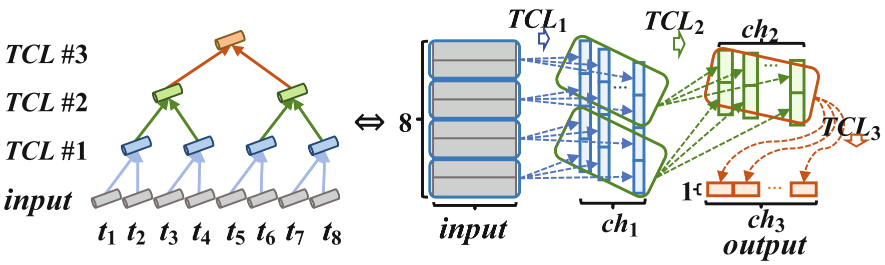


Fig. 14 TCP module for pseudo 3D′ feature extraction

4.3 Joint Loss Optimization

After feature fusion, we tackle long-range sequential learning. First, we discuss the CTTR module which generates the predicted sentence $\pi = \{\pi_n\}_{n=1}^N$. In order to further improve the temporal correlation and classification accuracy of features, we construct a pseudo-supervised learning framework. Here, $\pi = \{\pi_n\}_{n=1}^N$ is regarded as pseudo-labels of words for alignment learning of sequential features. According to the pseudo-labels, the FCLS module and the FCOR module respectively calculate the entropy of feature classification and the similarity of feature samples with the same or different labels. Both of the above two modules prevent the CTTR module from overfitting the training data. We combine the three loss functions $\mathcal{L}_{fcor}$, $\mathcal{L}_{cttr}$, and $\mathcal{L}_{fcls}$ to jointly optimize the model. The entire loss function is expressed as Eq. 24.

$$\mathcal{L} = \frac{1}{|\mathcal{S}|}\sum \mathcal{L}_{cttr} + \frac{1}{|\mathcal{M}|}\sum \mathcal{L}_{fcls} + \frac{1}{|\mathcal{T}|}\sum \mathcal{L}_{fcor} \tag{24}$$

where $\mathcal{S}$, $\mathcal{M}$ and $\mathcal{T}$ respectively represent the set of training samples, the set of clip-level features and the set of all pseudo triplets. $\mathcal{L}_{cttr}$ is expressed in Eq. 25, while $\mathcal{L}_{fcls}$ and $\mathcal{L}_{fcor}$ will be given in Eqs. 26 and 28.

4.3.1 CTC Loss for CTTR Module

The CTTR module is composed of a BGRU and CTC-decoder. BGRU is adopted to encode long-term temporal hints of visual features. Then, a fully connected layer embeds the embedding sequential features into non-normalized categorical probabilities $\{p_n\}$. Following the same operations as Eqs. 16 ∼ 18 in Sect. 3.3.1 of DenseTCN, we take the CTC loss as the objective function of the CTTR module, as shown in Eq. 25.

$$\mathcal{L}_{cttr} = -\sum_{\pi \in \mathcal{B}^{-1}(\mathcal{Y})} \sum_{n \in N} log(p_n^{\pi_n}) \tag{25}$$

where $\mathcal{B}^{-1}(\mathcal{Y})$ represents the set of alignments from the decoding path π to the target sentence $\mathcal{Y}$. Note that the CTC decoder selects the word label with the maximum value in each probability vector p_n to compose the generated sentence, which are regarded as pseudo-labels for the loss calculation in the following FCLS and FCOR modules.

4.3.2 Cross-Entropy Loss for FCLS Module

We use an FC layer with batch normalization and softmax to realize the FCLS module. This module obtains the predicted probability of each clip-level feature from the fusion feature $\{\boldsymbol{f}_n\}$. Based on the assumption that the pseudo-label is reliable, the optimization goal of the network is to make the new predicted probabilities trend toward the pseudo-labels. We calculate the cross-entropy loss to evaluate the distance between the predictions and the pseudo labels as Eq. 26.

$$\mathcal{L}_{fcls}(\mathcal{M}) = -\sum_{\mathbf{m}\in\mathcal{M}} \sum_{k\in K} \mathbf{y}_{\mathbf{m}}^k \log(p_{\mathbf{m}}^k) \tag{26}$$

where $p_{\mathbf{m}}^k$ represents the predicted probability of the sample $\mathbf{m}$ calculated by the FCLS module. $\mathbf{y}_{\mathbf{m}}^k$ is a boolean value indicating whether the label of the sample $\mathbf{m}$ is $\pi_{\mathbf{m}}$ according to the output of CTTR.

4.3.3 Triplet Loss for FCOR Module

We design a triplet loss to promote the features with the same label to gather in the embedding space, while the features with different labels are far away from each other. In the FCOR module, FC with batch normalization is used to model features into the embedding space to obtain new features $\{\boldsymbol{f}_n\}$.

According to the word decoding path π with the maximum probability of P_{max}^{π}, the embedding features are divided into different groups. Specifically, clip features with the same label are classified as positive pairs, denoted as (e_+, e_+); while clip features with different labels are regarded as negative pairs, denoted as (e_+, e_-) and (e_-, e_+). Thus, the set of all sample pairs can be represented as $\mathcal{T} = \{(e_+, e_+), (e_+, e_-), (e_-, e_+)\}$. Taking the video shown in Fig. 15 as an example, e_1 and e_6 are (e_+, e_+), while e_1 and e_2 are (e_+, e_-) or (e_-, e_+). We use colored squares and gray squares in the matrix to represent positive and negative pairs, respectively. Here, we ignore self-pairs (along the diagonal) and pairs with blank label samples (squares with snowflake dots). The constraint of the distance of positive and negative pairs is formulated in Eq. 27.

$$\begin{cases} s(e_+, e_+) > s(e_+, e_-) + \alpha \\ s(e_+, e_+) > s(e_-, e_+) + \alpha \\ s.t. \quad s(a, b) = \frac{a^T b}{\|a\|\|b\|} = L_2(a)^T \cdot L_2(b) \end{cases} \tag{27}$$

where $s(a, b)$ represents the similarity function with samples a and b, and L_2 denotes L_2-normalization. We use parameter α to control similarity intensity during the training process.

Then, we implement the measurement of triplet loss. For each batch, we randomly select the same number of negative pairs as positive pairs, and perform the triplet loss calculation as follows:

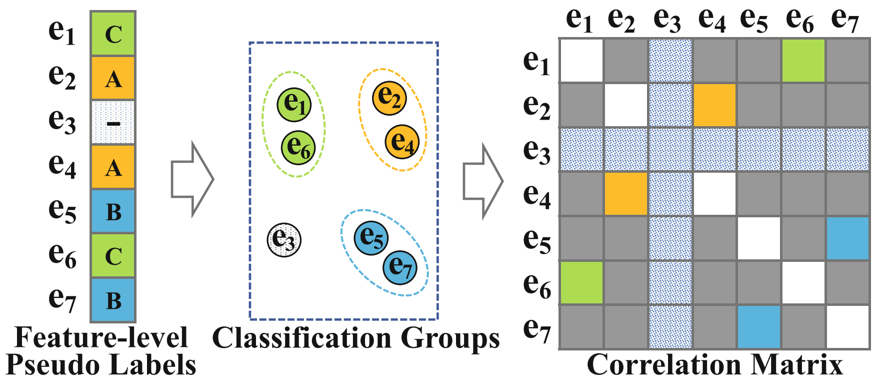


Fig. 15 Calculation of triplet loss with different labels. e_3 indicates the blank label

$$\begin{aligned}\mathcal{L}_{fcor}(\mathcal{T}) &= \sum_{t\in\mathcal{T}} max(s(t_{neg}) - s(t_{pos}) + \alpha, 0) \\ &= \sum_{t\in\mathcal{T}} max(s(t_{neg}) - \beta, 0) + \sum_{t\in\mathcal{T}} max(\beta - s(t_{pos}), 0)\end{aligned} \tag{28}$$

where β controls similarity intensity in the training process.

4.4 *Experiment*

4.4.1 Experiment Setup

Dataset and Evaluation We conduct experiments on RWTH-PHOENIX-Weather (PHOENIX) [16] and USTC-CSL—Split II [14] datasets for unseen sentences test. We adopt Word Error Rate (**WER**) [18] as the evaluation metric, which measures the minimal cost of insertion, deletion, and replacement operations during sequence conversion. For a sequence of length $|L|$, the proportions of inserted and deleted words are recorded as **ins** and **del**, respectively.

Implementation The resolutions of sign videos in the PHOENIX and USCT-CSL datasets are 210 × 260 pixels and 1280 × 720 pixels, respectively. We crop out the area covering the human body from the original frames and resize them to 224×224 pixels. We extract features with four-frames overlapping, and parameter setting of the TCP module are shown in Table 16, where the parameter $d_0 = d_1 = d_2 = d_3 = 512$-dim. We adopt the ADAM optimizer with batch size 40, and set the initial learning rate to 1×10^{-4}. The learning rate is reduced by $1/10$ every 20 epochs until the learning rate becomes 1×10^{-6}. ReLU and Dropout operations are adopted after each TCL layer, and the Dropout parameter is set to 0.2.

Table 16 Parameter setting of the TCP module

Input size	Layer	Kernel, channel, stride	Output size
$t \times d_0$	TCL_1	$2 \times d_0,\ d_1,\ 2$	$(t/2) \times d_1$
	TCL_2	$2 \times d_1,\ d_2,\ 2$	$(t/4) \times d_2$
	TCL_3	$2 \times d_2,\ d_3,\ 2$	$(t/8) \times d_3$

Table 17 Performance comparison on the PHOENIX dataset with different features and $\mathcal{L}_{cttr}$ loss

	VAL(%)	TEST (%)
Features	del/ins/WER	del/ins/WER
$\boldsymbol{f_{2d}}$	55.1/1.5/69.4	53.6/1.8/68.3
$\boldsymbol{f'_{3d}}$	27.5/5.8/63.6	26.8/6.1/62.2
$\boldsymbol{f_{3d}}$	21.0/5.1/45.1	20.0/5.5/45.4
$\boldsymbol{f'_{3d}} + \boldsymbol{f_{3d}}$	10.5/7.3/42.2	10.8/7.8/42.2
$\boldsymbol{Fusion_{\{f'_{3d}, f_{3d}\}}}$	10.6/6.9/**41.0**	10.1/7.9/**41.3**

Table 18 Performance comparison using different losses on PHOENIX dataset

	VAL (%)	TEST (%)
Loss	del/ins/WER	del/ins/WER
$\mathcal{L}_{cttr}$	10.6/6.9/41.0	10.1/7.9/41.3
$\mathcal{L}_{cttr}$+$\mathcal{L}_{fcls}$	10.2/6.7/39.9	10.3/7.7/40.2
$\mathcal{L}_{cttr}$+$\mathcal{L}_{fcor}$	11.3/6.7/39.8	10.9/6.9/40.0
$\mathcal{L}_{cttr}$+$\mathcal{L}_{fcls}$+$\mathcal{L}_{fcor}$	11.8/5.9/**38.9**	10.6/6.1/**38.7**

4.4.2 Model Validation

To verify the TCP module, we conducted experiments with different features. As shown in Table 17, the **del** value of $\boldsymbol{f_{2d}}$ is 55.1%, which is much worse. TCP improves the performance with 27.5%. Compared to direct addition of $\boldsymbol{f_{3d}}'$ and $\boldsymbol{f_{3d}}$, MLP fusion has better performance, and the WER value is reduced from 42.2% to 41.3% on the TEST set. We test the effectiveness of the pseudo-supervised learning framework. As shown in Table 18, the introduction of $\mathcal{L}_{fcls}$ improves the performance by 1.1% on both VAL and TEST sets. With the auxiliary of $\mathcal{L}_{fcor}$, the model learns the similarities and differences between clip-level representations, thereby further reducing the WER on the VAL and TEST sets by 1.0% and 1.5%. When using $\mathcal{L}_{cttr}$, $\mathcal{L}_{fcls}$, and $\mathcal{L}_{fcor}$ simultaneously, CTM can achieve the best results (Fig. 16).

4.4.3 Main Comparison

Here, the proposed model is compared to the state-of-the-art. We observe two obvious conclusions from Table 19. (1) The previous work always introduces extra visual hints to improve the performance, such as using visual representations of face or hands, and pose trajectory. In addition, the pre-trained sign language vocabulary is utilized in 1M-Hands [18], while CNN-Hybrid [17] introduces additional supervision. In contrast, our method has no extra hints and additional supervision. (2) Most methods use offline iterations, such as 1-M-H, CNN-Hybrid, Staged-Opt [5], and DCNN [24]. Without offline optimizations, our CTM achieves

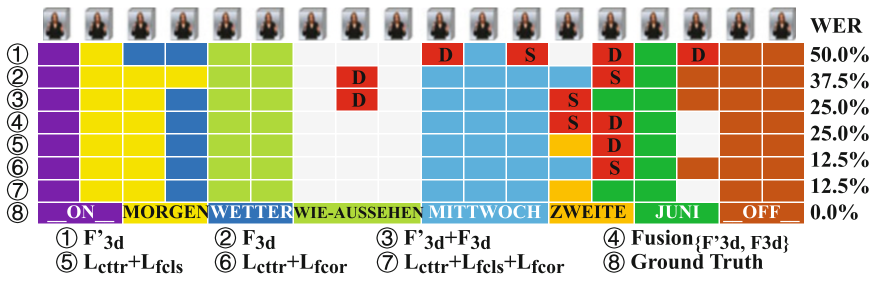


Fig. 16 Sentence decoding results using different setting of features and losses. "S" and "D" represent substitution and deletion operations

Table 19 Comparison with the state-of-the-art on the PHOENIX dataset

	Off-line	Extra	VAL (%)	TEST (%)
Methods	Iterations	Augmentation	des/ins/WER	des/ins/WER
HOG-3D [16]	–	√	25.8/4.2/60.9	23.2/4.1/58.1
CMLLR [16]	–	√	21.8/3.9/55.0	20.3/4.5/53.0
1-M-H [18]	3	√	19.1/4.1/51.6	17.5/4.5/50.2
1-M-H+CMLLR [18]	3	√	16.3/4.6/47.1	15.2/4.6/45.1
CNN-Hybrid [17]	3	√	12.6/5.1/38.3	11.1/5.7/38.8
Staged-Opt-init [5]	–	√	16.3/6.7/46.2	15.1/7.4/46.9
Staged-Opt [5]	3	√	13.7/7.3/39.4	12.2/7.5/38.7
SubUNets [1]	–	√	14.6/4.0/40.8	14.3/4.0/40.7
DCNN-init [24]	–		18.5/2.6/60.3	18.1/2.8/59.7
DCNN [24]	**5**		**8.3/4.8/38.0**	**7.6/4.8/37.3**
Our Method	–		**11.6/6.3/38.9**	**10.9/6.4/38.7**

"Extra Augmentation" refers to the date of face, hand or trajectory. "Off-line Iterations" represents the number of iterations during offline optimization, and "-" indicates the framework was trained in an end-to-end manner

better performance than HOG-3D [16], CMLLR [16], SubUNets [1], Staged-Opt-init and DCNN-init. It is worth noting that Staged-opt and DCNN obtain good performance through offline iterative optimization. Their initial results of WER reduce rapidly to 46.9% and 59.7% on the test dataset, which are much lower than the values obtained by our approach. Note that offline iteration requires repeated training. The optimization process is time-consuming. Our CTM model does not suffer from this limitation and achieves comparable performance to these offline iterative models.

From the results of Table 20, the proposed CTM achieves the best performance on the USTC-CSL dataset with performance improvements of 2.2∼5.1% compared with other models. Different from S2VT and HLSTM based on the encoder-decoder framework, CTM directly utilizes connectionist temporal modeling along the temporal dimension, which is also more flexible for online SLT.

Table 20 Comparison with the state-of-the-art on the USTC-CSL dataset

Methods	TEST WER (%)
S2VT [28]	67.0
S2VT(3-layer) [33]	65.2
HLSTM [9]	66.2
HLSTM-attn [9]	64.1
Our method	**61.9**

Acknowledgments This work is supported in part by the State Key Development Program under Grant 2018YFC0830103, in part by the National Natural Science Foundation of China (NSFC) under Grant 61876058, and in part by the Fundamental Research Funds for the Central Universities.

References

1. Camgoz, N.C., Hadfield, S., Koller, O., Bowden, R.: Subunets: End-to-end hand shape and continuous sign language recognition. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp. 3075–3084 (2017)
2. Celebi, S., Aydin, A.S., Temiz, T.T., Arici, T.: Gesture recognition using skeleton data with weighted dynamic time warping. In: VISAPP, pp. 620–625 (2013)
3. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets (2014). Preprint. arXiv:14053531
4. Cui, R., Liu, H., Zhang, C.: Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In: CVPR, pp. 7361–7369 (2017)
5. Cui, R., Liu, H., Zhang, C.: Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1610–1618 (2017)
6. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**(5814), 972–976 (2007)
7. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: International Conference on Machine Learning (ICML), pp. 369–376 (2006)
8. Guo, D., Zhou, W., Li, H., Wang, M.: Online early-late fusion based on adaptive HMM for sign language recognition. TOMCCAP **14**(1), 1–18 (2017)
9. Guo, D., Zhou, W., Li, H., Wang, M.: Hierarchical LSTM for sign language translation. In: AAAI, pp. 6845–6852 (2018)
10. Guo, D., Tang, S., Wang, M.: Connectionist temporal modeling of video and language: a joint model for translation and sign labeling. In: IJCAI, pp. 751–757 (2019)
11. Guo, D., Wang, S., Tian, Q., Wang, M.: Dense temporal convolution network for sign language translation. In: IJCAI, pp. 744–750 (2019)
12. Guo, D., Zhou, W., Li, A., Li, H., Wang, M.: Hierarchical recurrent deep fusion using adaptive clip summarization for sign language translation. ACM TIP **29**, 1575–1590 (2020)
13. Hara, K., Kataoka, H., Satoh, Y.: Learning spatio-temporal features with 3d residual networks for action recognition. In: ICCV Workshop on Action, Gesture, and Emotion Recognition, vol. 2, p. 4 (2017)
14. Huang, J., Zhou, W., Zhang, Q., Li, H., Li, W.: Video-based sign language recognition without temporal segmentation (2018). Preprint. arXiv:180110111
15. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **20**(3), 226–239 (1998)

16. Koller, O., Forster, J., Ney, H.: Continuous sign language recognition: towards large vocabulary statistical recognition systems handling multiple signers. Comput. Vision Image Understanding **141**, 108–125 (2015)
17. Koller, O., Ney, H., Bowden, R.: Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3793–3802 (2016)
18. Koller, O., Ney, H., Bowden, R.: Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In: CVPR, pp. 3793–3802 (2016)
19. Koller, O., Zargaran, O., Ney, H., Bowden, R.: Deep sign: hybrid cnn-hmm for continuous sign language recognition. In: British Machine Vision Conference (BMVC), p. 12 (2016)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS), pp. 1097–1105 (2012)
21. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: EMNLP, pp. 1412–1421 (2015)
22. Pan, Y., Mei, T., Yao, T., Li, H., Rui, Y.: Jointly modeling embedding and translation to bridge video and language. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4594–4602 (2016)
23. Pei, X., Guo, D., Zhao, Y.: Continuous sign language recognition based on pseudo-supervised learning. In: MAHCI, pp. 33–39 (2019)
24. Pu, J., Zhou, W., Li, H.: Dilated convolutional network with iterative optimization for coutinuous sign language recognition. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 885–891 (2018)
25. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intell. Data Anal. **11**(5), 561–580 (2007)
26. Song, P., Guo, D., Zhou, W., Wang, M., Li, H.: Parallel temporal encoder for sign language translation. In: ICIP, pp. 1915–1919 (2019)
27. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp. 4489–4497 (2015)
28. Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., Saenko, K.: Sequence to sequence-video to text. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 4534–4542 (2015)
29. Wang, J., Liu, Z., Wu, Y., Yuan, J.: Mining actionlet ensemble for action recognition with depth cameras. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1290–1297 (2012)
30. Wang, H., Chai, X., Zhou, Y., Chen, X.: Fast sign language recognition benefited from low rank approximation. In: Automatic Face and Gesture Recognition (FG), vol. 1, pp. 1–6 (2015)
31. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: towards good practices for deep action recognition. In: ECCV, pp. 20–36 (2016)
32. Wang, S., Guo, D., Zhou, W., Zha, Z., Wang, M.: Connectionist temporal fusion for sign language translation. In: ACM MM, pp. 1483–1491 (2018)
33. Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., Courville, A.: Describing videos by exploiting temporal structure. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 4507–4515 (2015)
34. Zhang, J., Zhou, W., Xie, C., Pu, J., Li, H.: Chinese sign language recognition with adaptive HMM. In: International Conference on Multimedia and Expo (ICME), pp. 1–6 (2016)
35. Zheng, L., Wang, S., Tian, L., He, F., Liu, Z., Tian, Q.: Query-adaptive late fusion for image search and person re-identification. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1741–1750 (2015)
36. Zhu, W., Hu, J., Sun, G., Cao, X., Qiao, Y.: A key volume mining deep framework for action recognition. In: CVPR, pp. 1991–1999 (2016)