Hierarchical LSTM for Sign Language Translation

Dan Guo[1], **Wengang Zhou**[2], **Houqiang Li**[2], **and Meng Wang**[1]
[1]School of Computer and Information, Hefei University of Technology, Hefei, 230009, China
[2]EEIS Department, University of Science and Technology of China, Hefei, 230027, China
guodan@hfut.edu.cn, zhwg@ustc.edu.cn, lihq@ustc.edu.cn, eric.mengwang@gmail.com

Abstract

Continuous Sign Language Translation (SLT) is a challenging task due to its specific linguistics under sequential gesture variation without word alignment. Current hybrid HMM and CTC (Connectionist temporal classification) based models are proposed to solve frame or word level alignment. They may fail to tackle the cases with messing word order corresponding to visual content in sentences. To solve the issue, this paper proposes a hierarchical-LSTM (HLSTM) encoder-decoder model with visual content and word embedding for SLT. It tackles different granularities by conveying spatio-temporal transitions among frames, clips and viseme units. It firstly explores spatio-temporal cues of video clips by 3D CNN and packs appropriate visemes by online key clip mining with adaptive variable-length. After pooling on recurrent outputs of the top layer of HLSTM, a temporal attention-aware weighting mechanism is proposed to balance the intrinsic relationship among viseme source positions. At last, another two LSTM layers are used to separately recurse viseme vectors and translate semantic. After preserving original visual content by 3D CNN and the top layer of HLSTM, it shortens the encoding time step of the bottom two LSTM layers with less computational complexity while attaining more nonlinearity. Our proposed model exhibits promising performance on singer-independent test with seen sentences and also outperforms the comparison algorithms on unseen sentences.

Introduction

This paper studies the problem of vision-based Sign Language Translation (SLT), which bridges the communication gap between the deaf mute and normal people. It is related to several video understanding topics that targets to interpret video into understandable text and language. To be specific, SLT is derived from single sign word recognition, which is a kind of action recognition or video classification (Cai et al. 2016). But in normal dialogues (i.e., sentence videos), continuous sign dynamics usually lack word alignment. It is difficult to precisely align sub-video clips to each sign word and recognize them. Then on the other hand, SLT is similar to video captioning in that the video is directly translated to text sequences (Yao et al. 2015). The primary difference lies in that video captioning uses grammar knowledge and semantics coherence with feature representation of object(s), scene, motion or action for sentence generation, while SLT emphasizes on semantic understanding with word-to-word transition among independent sub-video clips. SLT suffers the specific linguistics challenges, such as adverb modifying verb, e.g., the adverb in phase "run quickly" is signed by speeding the word "run" up; uncertain directional verbs and positional signs between the signer and referent(s); unknown words with finger spelling; non-hand features, e.g., facial expression and lip shape; etc. Learning visual word in SLT is much more strict.

Our problem belongs to weakly supervised learning with the lack of supervision on accurate temporal locations for sign words. Several recent works made some success to solve frame-level or gloss(word)-level alignment, such as hybrid HMM and CTC embedded into deep learning (Koller, Zargaran, and Ney 2017; Cui, Liu, and Zhang 2017). However, they are limited to a prerequisite that word label in sentences is consistent to the order of corresponding visual content. If the word order is reversed or messed as in NMT (Neural Machine Translation, e.g. from English to French (Bahdanau, Cho, and Bengio 2014)), they are unsuitable to tackle sequential frame-level classification under word labels in disorder for text generation.

Our work is free of such limitation. We adopt the encoder-decoder framework, which respectively learns visual content and word embedding. More importantly, to effectively encode the visual semantic, a Hierarchical LSTM-based (HLSTM) model is proposed. The key idea of our model is to build a multi-layered visual-semantic embedding architecture with different granularities, i.e., frames, visemes (sub-visual-word), visual-words and the entire video. Our presupposition is tackling visual feature embedding of sub-sign units, i.e., visemes. We seek for the high level representation of visemes and focus on the transition among these visemes to avoid directly bridging the whole video frames and natural language. In other words, our model is characterized with a hierarchical and variable-sized temporal structure, which explores the spatio-temporal cues for variable-sized chips of visemes.

The framework of our approach is illustrated in Figure 1. A 3D CNN model (i.e., C3D (Tran et al. 2015)) is firstly utilized to extract visual features. It's because that in our dataset, a sentence video often has many successive original frames. Here, 3D CNN is more useful than 2D CNN to learn

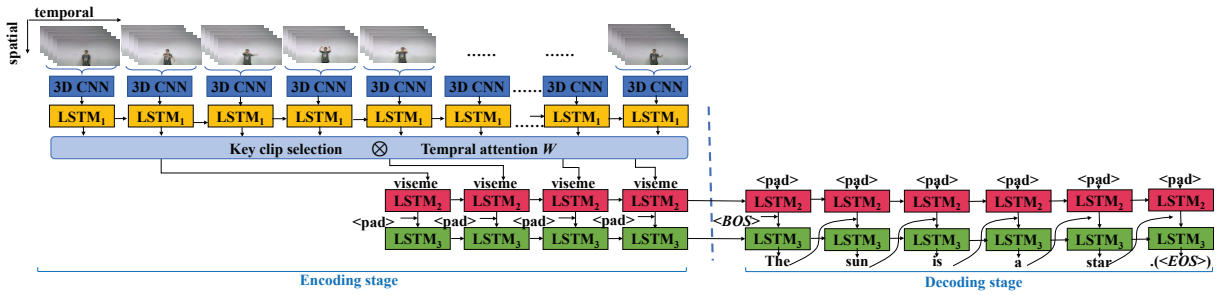

Figure 1: The general architecture of HLSTM. In our end-to-end model, we take the pre-trained C3D model for 3D CNN feature extraction, and key clip mining is an online heuristic algorithm without deep training. It uses a zero-padding vector as the beginning-of-sentence tag $<BOS>$ to start, and terminates until it meets the ending symbol (.), namely $<EOS>$.

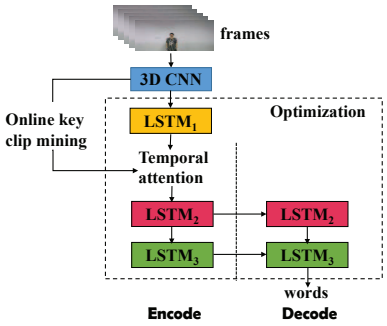

Figure 2: The basic modular architecture of HLSTM.

spatio-temporal context, and it can avoid dependency attenuation of long sequence transmission in the LSTM learning. Secondly, signs can be easily recognized by discriminative gesture or upper-body skeleton variations that sparsely hide in the videoes. To distinguish key (discriminative) and less-important clips, we utilize an online adaptive key clip mining method by optimizing residual square sum of previous successive 3D CNN features and current feature, and capture their linear correlation. The motivation is to avoid over-training with less-important clips that may degrade performance.

Thirdly, to weaken less-important clips, three pooling strategies are proposed to further capture the recurrent characteristics of visemes by $LSTM_1$ (the top LSTM layer of the model). Besides, an attention-aware weighting mechanism is proposed in the encoding stage along the time dimension. It balances the intrinsic relationship among source positions related to the entire translated sentence. Finally, when feature sequences are exhausted in encoding stage, they arrive at the decoding stage. We respectively implement visual and word embedding with $LSTM_2$ and $LSTM_3$ (the bottom two LSTM layers of the model). $LSTM_2$ is served for visual representation in encoding stage and $LSTM_3$ is used to model word representation in decoding stage for sequence learning.

In a nutshell, we present a hierarchical encoder-decoder framework to solve continuous SLT. Our end-to-end HLSTM model is a mixture of CNN and RNN. It explores visemes by online adaptive variable-length key clip segmentation and temporal attention. With the mixed transition between viseme representations, HLSTM achieves a high-level visual-semantic embedding learning.

Related work

This section reviews related work to SLT in three aspects: hand-crafted feature, traditional recognition model, and current prevalent deep feature and model.

Hand-crafted feature: Traditional feature representation for SLT include depth based feature and visual feature (Pu, Zhou, and Li 2016). Popular depth feature contains point clouds (Wang et al. 2012) and surface normals (Lin et al. 2014). However, due to high cost and inconvenience of calibration, depth sensors and auxiliary devices are unpopular. Meanwhile, more and more researches focus on vision-based recognition systems which only consider the associated RGB channels. Turning to visual feature, Hernandez-Vela et al. presented the Bag-of-Visual-and-Depth-Words for multimodal feature fusion (Hernández-Vela et al. 2012). Tewari et al. employed the AdaBoost algorithm based on HAAR-like features to integrate weaker classifiers into a strong classifier (Tewari et al. 2015). HOG feature is also applied for sign language recognition (Wang et al. 2015).

Traditional recognition model: As for isolated sign or gesture classification, the SVM model was applied. For instance, the Grassmann covariance matrix (GCM) model was employed as the kernel of SVM classifier (Wang et al. 2016a). At the same time, temporal context were also considered to acquire sequential dynamics. Celebi et al. developed a weighted Dynamic Time Warping (DTW) to optimize the discriminant ratio of joints between two temporal sequences (Celebi et al. 2013). Similar features can be grouped into clusters or blocks, termed as states. The state transition is modeled using probability distribution or graphical models, e.g. various Hidden Markov Models (HMM) (Yang and Sarkar 2006; Guo et al. 2016; 2017), Hidden Conditional Random Fields (HCRFs) (Wang et al. 2006), Autoregressive Models (AR) (Ishihara and Otsu 2004), etc. Among them, HMM is the most widely known.

Deep feature & model: Compared to above traditional work, deep learning approaches have achieved impressed success in computer vision. They are also prevalent in sign

language and gesture recognition, such as CNN (Huang et al. 2015; Camgoz et al. 2016), LSTM (Liu, Zhou, and Li 2016), RNN, etc. For instances, a CNN-based multi-scale framework was proposed to detect gesture (Neverova et al. 2016). Molchanov et al. designed a 3D CNN extractor to jointly represent both appearance and motion variation (Molchanov et al. 2016). Two stream RNNs is designed to fuse multi-modal features(Chai et al. 2015). Lefebvre et al. proposed a bidirectional LSTM-RNN to tackle the preceding and following context of dynamic variation (Lefebvre et al. 2015).

The prevalent trend is to mix merits of CNN feature extractor and sequential learning models (i.e., RNN and LSTM, even HMM), such as deep DNN & HMM (DDNN) (Wu et al. 2016), temporal convolutions & bidirectional RNN (Pigou et al. 2015), recurrent 3D convolutional neural network (R3DCNN) (Molchanov et al. 2016), etc. These mixture approaches effectively model both spatial and temporal context of motion variation. But most of them are designed for isolated gesture or sign word recognition and cannot be readily used for continuous sign sentence translation.

Then temporal boundaries of sign words in continuous SLT have been extensively studied. There are sign spotting and word alignment analysis, such as hybrid HMM and CTC embedded into deep learning. Different from sign spotting, our problem belongs to weakly supervised learning with the lack of supervision on accurate temporal separation for sign words. As for word alignment, Koller et al. embedded deep CNN into the HMM framework to solve frame-level alignment (Koller, Zargaran, and Ney 2017). Cui et al. proposed an LSTM & CTC to solve the gloss(word)-level alignment (Cui, Liu, and Zhang 2017). As mentioned in introduction, the same prerequisite of these work is that word order in sentences should be consistent with that of corresponding visual clips. If the word is out of order to its corresponding visual content, they are unsuitable for SLT. In contrast, our work is not limited to this with the encoder-decoder framework.

Our Proposed Approach

We implement viseme-unit encoding and text decoding by neural network models, where the input is the sequence of video frames $(f_1, \cdots, f_N)$, and the output is text translation, namely a sequence of text words $(y_1, \cdots, y_m)$. In the model, we employ pre-trained C3D (Tran et al. 2015) model to obtain convolutional features. Then we segment key and less-important sub-video clips by an online heuristic algorithm. Next, after imposing pooling and attention-aware mechanisms, our HLSTM encoder compacts less-important video clips and summarizes key clips into a high level recurrent representation (viseme vectors), and finally the decoder stage outputs a sentence with a variable length. Key clip mining and the HLSTM encoder are introduced below.

Online Key Clip Mining

Discriminative motion modalities or patterns always sparsely hide among the whole video, which occur irregularly under different conditions, such as speed, habit of signer, and special constituents of sign word, etc. Different from extracting key frames or volumes with fixed time interval (Wang et al. 2016b; Zhu et al. 2016), here we adopt an online mining approach to automatically obtain variable-length key clips. We use the low rank approximation method in (Wang et al. 2015) to get the linear correlation of consecutive frame stream. The idea is to calculate the feature residual sum of square (RSS) ϵ between previous frames and current frame.

Given video feature stream $F = [\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_n]$, we use the correlation matrix M to calculate the residual error ϵ_c at current feature $\boldsymbol{f}_c$. The subset of F from $\boldsymbol{f}_1$ to $\boldsymbol{f}_c$ is denoted as $F_c = [\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_c]$. The Initialization are $\epsilon_1 = 0$ and $M = (\boldsymbol{f}_1^T \boldsymbol{f}_1)^{-1}$. At time step c, $2 \leq c \leq n$, we compute the correlation coefficient $\boldsymbol{\beta}_c$ and residual error ϵ_c as follows:

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

Next we update the core matrix for next feature $\boldsymbol{f}_{c+1}$:

$$M = \begin{bmatrix} M + \boldsymbol{\beta}_c^T \boldsymbol{\beta}_c \epsilon_c & -\boldsymbol{\beta}_c / \epsilon_c \\ -\boldsymbol{\beta}_c^T \epsilon_c & 1/\epsilon_c \end{bmatrix} \tag{2}$$

where M summarizes the intrinsic linear correlation of the feature set F_c, $\boldsymbol{\beta}_c$ establishes the mapping relationship of F_{c-1} and $\boldsymbol{f}_c$ (i.e., the relevant weight on each previous frame by the measure M), and $F_{c-1}\boldsymbol{\beta}_c$ is the approximate reconstruction of $\boldsymbol{f}_c$ by utilizing F_{c-1} at current time c. Finally, we obtain $\epsilon = [\epsilon_1, \cdots, \epsilon_c, \cdots, \epsilon_n]$.

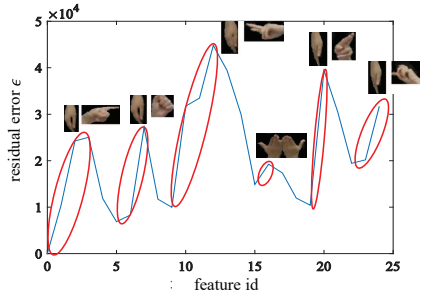

Figure 3: Curve of RSS variable ϵ of a video. Each peak point corresponds to a discriminative gesture of sign words.

Continuous variation of key clips are effective to mine real motion patterns in SLT. Different from obtaining the optimal subset of selected discrete frames in previous work, the RSS curve is helpful to segment the key and less-important clips with variable length (e.g., invalid fragment of the word-to-word transition). As observed in Figure 3, each peak point in the residual error curve indicates the local maximal gain of accumulative error of consecutive variation. We remain the monotonic increasing part of the curve of residual error ϵ as the profits, as it cannot be replaced by previous frames with the continuously increasing residual error. After that, ϵ drops gradually along the monotonic decreasing part of the curve. This means that the decreasing part could be linearly reconstructed by previous frames with downward error.

We select those frames in each monotonically increasing part of the residual error curve as **a key clip** and weaken the negative effect of those frames in the monotonic decreasing

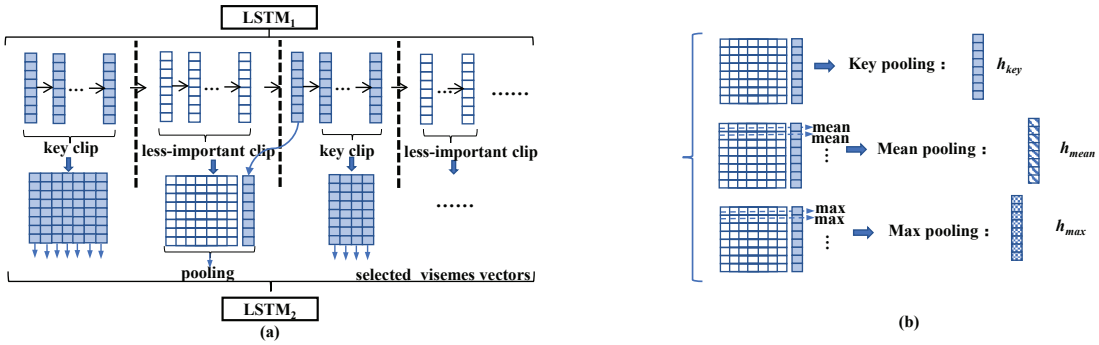

Figure 4: Illustration of compacting less-important clips by pooling strategy. (a). The pooling process. (b). Different pooling strategies.

Table 1: 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 $\mathscr{G}(\cdot)$
n''	the number of encoding time steps of $LSTM_2$.

parts that are deemed as **less-important clips**. Our motivation is to avoid over-learning less-important clips, which will prevent us to discover accurate sign modalities. The summarizing less-important clips strategy is detailed in HLSTM encoder.

Hierarchical LSTM Encoder

As shown in Figure 1, our HLSTM model is asymmetrical. Besides that the encoding and decoding stages have different lengths, i.e., a large number of frames versus a few words, its encoding layers are variable-length. HLSTM aims at a compact and effective visual representation for sign linguistics.

3-layer encoder The encoder uses both CNN and LSTM modules to encode the input video frames $(f_1, \cdots, f_N)$ into a visual embedding representation V:

$$\begin{aligned} V &= \psi_{lstms}[\mathscr{G}(\psi_{cnn}(f_1, \cdots, f_N))] \\ &= \psi_{lstms}[\mathscr{G}(\boldsymbol{f}_1, \boldsymbol{f}_1, \cdots, \boldsymbol{f}_n)] \\ &= \psi_{LSTM_3, LSTM_2, LSTM_1}[(\boldsymbol{f}'_1, \cdots, \boldsymbol{f}'_{n'})] \\ &= \psi_{LSTM_3, LSTM_2}(\widetilde{h}_1, \cdots, \widetilde{h}_{n''}) \\ &= (v_1, \cdots, v_{n''}) \end{aligned} \tag{3}$$

where $\mathscr{G}(\cdot)$ denotes the key clip mining. As noted in Table 1, N, n and n' are variable lengths. In this paper, we denote the average number of features of all training videos as l_{ave} and set $n'' = l_{ave}$. $\{\widetilde{h}_1, \cdots, \widetilde{h}_{n''}\}$ is the inputs for $LSTM_2$ detailed in the following descriptions of polling and attention-aware weighting mechanisms and $\{v_1, \cdots, v_{n''}\}$ is the hidden states of $LSTM_3$ on the encoding stage.

The HLSTM model is depicted in Figure 1 and 2 with a 3-layer LSTM encoder architecture. The top $LSTM_1$ is used to obtain recurrent representation based on 3D conventional

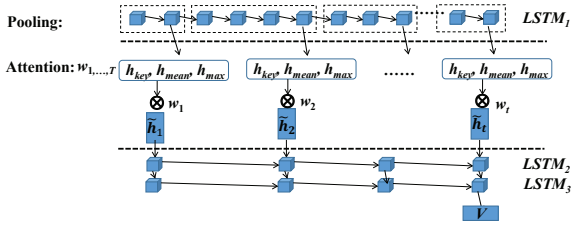

Figure 5: Attention-aware weighting mechanism.

features $F = [\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_n]$ extracted by the well-known C3D model (Tran et al. 2015). After imposing pooling strategy and attention-aware weighting, we shorten the recurrent features of $LSTM_1$ into the length-n''. Finally, we respectively implement the visual and word embedding with $LSTM_2$ and $LSTM_3$, where $LSTM_2$ is chiefly served for visual representation in encoding stage and $LSTM_3$ is used to model word representation in decoding stage for sequence learning. $LSTM_2$ and $LSTM_3$ are applied in both the encoding and decoding stages. This manifests that their parameters are shared in two stages.

Pooling strategy Figure 1 unrolls HLSTM over time. After $LSTM_1$, its outputs $\{h_t\}(t \in [1, n])$ have to be shortened for $LSTM_2$. As shown in Figure 4, if h_t belongs to a key clip, it is taken as viseme vector except for the first time step. If h_t belongs to a less-important clip, we conduct pooling on its less-important clip together with the first frame of the following adjacent key clip. We define the pooled feature chunk as $\{h_t\}$ $(t \in [t_1^*, t_{T_c}^*])$, where T_c is the sum of the time step length of the less-important clip and 1. There are three pooling strategies to weaken less-important clips as follows:

- **Key pooling:** The last step of the chunk is directly taken as the input $h'_t = h_{key}$ for $LSTM_2$. This pooling drops less-important clip's outputs, but keeps their gradually recurrent characteristic.
- **Mean pooling:** The average vector of the chunk along temporal dimension is taken as the input h'_t . Here $h'_t = h_{mean}$ averages the effect of recurrent outputs of the chunk.

- **Max pooling:** The maximization on the chunk along temporal dimension is taken as the input $h'_t = h_{max}$. Here h'_t highlights the prominent response of the chunk .

After pooling, we feed the recurrent output $\{h'_t\}$ ($t \in [1, n']$) into the fixed-length n''. If $n' < n''$, we fill with zero-padding vectors. If the length of $n' \geq n''$, we make systematic sampling on $\{h'_t\}$ to match the fixed length n''.

Attention-aware weighting Different from most of attention models describing the relevance of **so**urce positions (encoder's hidden states $\{h_{so}\}$) and **ta**rget position (current decoder's hidden state h_{ta}), our attention-aware mechanism weights each source position to the entire translated sentence ($\{h_{ta}\}$). It means that our attention balances the intrinsic relationship among source positions. It is depicted in Figure 5, which weights each source position's impact along time dimension. The matrix $W \in \mathbb{R}^{T_e}$ is automatically learned by training $\widetilde{h}_t = w_t \cdot h'_t$ in our end-to-end architecture, where h'_t is one kind of above three pooling outputs and $T_e = n''$. We will make evaluation with and without this module, denoted as HLSTM-attn and HLSTM, respectively.

Sentence Generation

Turning to the decoding stage, $LSTM_2$ and $LSTM_3$ are still kept for sentence generation. The motivation is to represent visual content and word at respective layer. Here $LSTM_3$ runs recursively to output word sequences with varying lengths. Given the representation V of the encoding stage, the decoder estimates the conditional probability of the output sequence $(y_1, \cdots, y_m)$:

$$p(y_1, \cdots, y_m|V) = \prod_{t=1}^{m} p(y_t|v_{n''+t-1}, y_{t-1}) \quad (4)$$

In the decoding stage, $LSTM_2$ feeds zero-padding vectors as empty visual inputs and $LSTM_3$ starts with the beginning tag (<*BOS*>) and sequentially feeds the word embedding vectors. During training, $LSTM_3$ inputs the embedding vector of previous ground truth word at each step. At test time, the output (z_t) of $LSTM_3$ is used to predict current word (y_t) with the maximum probability after the softmax function as shown in Equation 5. We select word (y_t) with the maximum probability in vocabulary and feed its word embedding vector into next time step.

$$p(y_t|z_t) = \frac{exp(W_y z_t)}{\sum_{z'_t=V} exp(W_y z'_t)} \quad (5)$$

In this paper, we use the entropy of generated sentence to learn the model parameter ψ. The loss optimization is only implemented during decoding stage while training. It maximizes the entropy by the log-likelihood of the predicted sentences of the entire training dataset using stochastic gradient descent. While this loss is propagated back in time, the model parameter ψ is updated, which is formulated as:

$$\psi^* = \underset{\psi}{\operatorname{argmax}} \sum_{t=1}^{m} p(y_t|v_{n''+t-1}, y_{t-1}; \psi) \log p(y_t|v_{n''+t-1}, y_{t-1}; \psi) \quad (6)$$

Table 2: Two splitting strategies for the 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	$50 \times 6 = 300$

Table 3: Comparison on different models using Split I

Model	**Precision**
LSTM&CTC (Warp-ctc)]	0.858
S2VT (Venugopalan et al. 2015)	0.897
LSTM-E (Pan et al. 2016)	0.882
LSTM-Attention (Yao et al. 2015)	0.851
LSTM-global-Attention (Luong, Pham, and Manning 2015)	0.858
HLSTM(SYS sampling)	**0.910**
HLSTM	**0.924**
HLSTM-atten	**0.929**
Model	**WER**
LSTM+CTC	0.119
HLSTM	**0.107**
HLSTM-atten	**0.102**

Experiment

Experiment Setup

Dataset: Our dataset is a collection of videos covering 100 daily sentences in Chinese sign language (CSL)[1]. Each sentence is played by 50 signers. It contains 50*100=5000 videos. The vocabulary size is 179. Each sentence contains 4∼8 (average 5) sign words (phases).

To validate our method, we split the dataset by two strategies shown in Table 2. (a) **Split I - signer independent test:** It splits video samples of 40 signers as training set and that of the remaining 10 signers as test set. The sentences of training and test sets are the same, but the signers are different. (b) **Split II - unseen sentences test:** This strategy elaborately selects up to 6 sentences as test set, and the left 94 sentences as training set. The split meets the constraint that words in the 6 sentences have separately appeared in the remaining 94 sentences, but each words's context, occurrence order and application scenarios are completely different.

Evaluation Metrics: We evaluate by the strict metric **precision**, namely the ratio of correct sentences. When generated sentence and the reference are completely the same, the generated sentence is deemed correct. In addition, we calculate the ratio of correct words to reference words in a sentence and denote the mean ratio as ***Acc-w***. Word error rate (WER) (Cui, Liu, and Zhang 2017) is used which measures the least number of operations to change translated sentence into the reference. We also adopt semantics evaluation metrics widely used in NLP, NMT, and image Description Evaluation, i.e., BLEU, METEOR, ROUGE-L and CIDEr.

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

Table 4: Feature comparison on Split I under $n_{hid} = 256$

Model	Feature	Precision
S2VT	VGG	0.196
	C3D	0.849
S2VT (3-layer)	VGG	0.398
	C3D	0.850
HLSTM	VGG	0.446
	C3D	0.852

Table 5: Precision comparison on different hidden-dims of LSTM under C3D features ($n'' = 21$, Split I)

	$n_{hid} = 256$	$n_{hid} = 512$	$n_{hid} = 1000$
S2VT	0.849	0.870	**0.897**
HLSTM	0.852	0.879	**0.924**

Comparison to Other Approaches

We compare HLSTM with the **LSTM&CTC** model[2], which is widely used in sequential data analysis such as speech recognition and sign language recognition. To be fair, we adopt the same features extracted by the pre-trained C3D and feed them into LSTM&CTC in our experiments. Meanwhile, as our work also belongs to the encoder-decoder paradigm, we compare HLSTM to some existing similar models. **S2VT** is a standard 2-layer stacked LSTM architecture with fixed encoder length (Venugopalan et al. 2015). **LSTM-E** inputs deep 2D or 3D CNN features with mean pooling for visual sematic embedding (Pan et al. 2016). **LSTM-Attention** embeds an attention mechanism to capture the temporal relationship among frames (Yao et al. 2015). **LSTM-global-Attention** explores a global attention mechanism for NMT (Luong, Pham, and Manning 2015). Besides, in the following experiments, if not specified, our HLSTM merely uses C3D features without temproal attention. HLSTM chooses mean pooling for Split I and max pooling for Spilt II. The reason is given in model validation. As for extension, HLSTM (SYS sampling) removes key clip selection and directly feeds the recurrent outputs of $LSTM_1$ into $LSTM_2$ by systematic sampling in HLSTM, and HLSTM-attn adds temporal attention to HLSTM.

As shown in Table 3, compared to LSTM&CTC, our model achieves better performance with higher precision and less WER. The LSTM&CTC framework aims at word-level alignment, which does not well learn word semantics. Compared to similar encoder-decoder frameworks, there are four aspects conclusions: (1) Compared to S2VT with the fixed-length stacked architecture, our hierarchical architecture achieves better performance. (2) LSTM-E implements average pooling on whole features, while our model pools on subunit chunk (less-important volumes). Experimental result indicates our pooling approach has better performance. (3) Classic attention mechanism which focuses on the relevance of source positions and current target position, does not work well on our datatset, such as LSTM-Attention and LSTM-global-Attention. These attentions repeatedly update each weight of source positions at each time, and spread the global influence to current target position. Differently, our attention strategy emphasizes accumulative weighting transition along the time dimension. It just remains transitive influence of source position until current position. For the SLT problem, the latter is much more feasible. (4) At last, among our HLSTMs, HLSTM (SYS sampling) is the worst, HLSTM is better while HLSTM-attn achieves the best performance, which can be attributed to the key clip selection and temporal attention-aware weighting.

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

Table 6: 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 7: 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**

Model Validation

As S2VT achieves the closest performance to our HLSTMs, we extend it to the S2VT(3-layer) and further compare in subsequent model validation experiments. S2VT(3-layer)'s encoder is set to 3-layer stacked LSTM with equal length.

Experiment on 2D and 3D CNN features: To compare 2D and 3D CNN features, we employ VGG and C3D models. VGG extracts features from each frame (Simonyan and Zisserman 2015). C3D extracts features from each video chunk clipped by every 16-frame with 8-frame overlap between two adjacent chunks (Tran et al. 2015). Therefore, the number of VGG features is 8 times of C3D. Limited by the high dimension (4096-dim) and a large number of VGG features, we experiment with the dimension of LSTM's hidden states $n_{hid} = 256$ to make calculation not out of memory. Table 4 shows that the C3D feature is obviously better than the VGG feature by either original S2VT, the extended 3-layer S2VT or HLSTM. C3D still has the advantage of action capturing for SLT. With the compact C3D features, the defect of gradient disappearance for long sequence learning is not serious. Thus the C3D model is taken as feature extractor in subsequent experiments.

Experiment on different LSTM hidden state numbers: To test the precision with different LSTM unit settings, we set n_{hid} to 256, 512 and 1000, respectively. As shown in Table 5, when the n_{hid} is set larger, the precision is raised. What's more, by our observation, when n_{hid} is small, experimental results are instable under multiple random tests.

Table 8: 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 sampling)	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 9: 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 sampling)	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

However, when $n_{hid} = 1000$, the results are stable. Thus we select the $n_{hid} = 1000$ as our LSTM parameter setting.

Experiment on different pooling strategies: From Table 6, the result accords with properties of these pooling strategies. To weaken less-important clip, along temporal dimension, key pooling keeps its recurrent characteristics, mean pooling averages its recurrent outputs, while maximal pooling highlights its prominent response. Therefore, mean pooling is the best to remember the average response of seen sentences. As for unseen sentence, max pooling is the best to retain maximum response of discriminative gestures of distinct sign words. Thus we separately set mean and max pooling for Split I and II. It is notable that key pooling makes some compromise with mediate performance.

Experiment on different n'' settings: Under our dataset, 66 is the maximum length of C3D features of a video under all training samples, and 21 is the mean length. Note that N, n and n' are variable lengths for different videos. Therefore, just n'' needs to be discussed. If $n'' = 66$, it recurrently induces all sequential features; and if n'' is set to 21, it means compacting features to feed into mean length. As shown in Table 7, better results are obtained with $n'' = 21$ than with $n'' = 66$. When $n'' = 66$, it makes little gain to supplement useless padding vectors on $LSTM_2$ and $LSTM_3$ layers. In other words, compact vector representation helps to achieve better performance.

Experiment on different encoder frameworks: As shown in Table 7, there are five similar but different encoding frameworks. We see that a fixed length of encoder is not a good choice, such as original S2VT and S2VT (3-layer). The hierarchical recurrent encoder works well. Bedsides, 3-layer is better than 2-layer as it incorporates recurrent characteristic by the top LSTM. At last, among HLSTMs, systematic sampling is worse than variable-length key clip mining. With temporal attention, HLSTM-attn achieves the best performance.

Summary of experiments on seen sentences: Table 8 concludes evaluation comparison. The performances of precision and semantic metrics are consistent. These results verify the strong capability of the HLSTM models on seen sentence under signer-independence test again.

Extended Experiment on Unseen Sentences

Generally, it is much more difficult for SLT on unseen sentences than on seen sentences. In the evaluation dataset, words in the 6 testing sentences have dispersedly appeared in the 94 sentences under totally different semantics context, occurrence order and application scenarios. Meanwhile, distribution of word samples in videos is extremely imbalanced. Even so, as shown in Table 9, HLSTMs still excels at recognizing more meaningful words than others.

Conclusions

We have proposed a hierarchical-LSTM framework for sign language translation, which builds a high-level visual-semantic embedding model for SLT. It explores visemes via online variable-length key clip mining and attention-aware weighting. The experiments show that our model achieves promising performance, especially under the independent tests for seen sentences with discriminative capability. But unseen sentence translation is still a challenging problem with limited sentence data (e.g., unbalance data distribution of common words and obscure words) and unsolved out-of-order word alignment. In next work we will make effort to explore adaptable word alignment for translation.

Acknowledgments

The work of W. Zhou was supported in part by NSFC under Contract 61472378 and Contract 61632019, in part by the Young Elite Scientists Sponsorship Program by CAST under Grant 2016QNRC001, and in part by the Fundamental Research Funds for the Central Universities.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Cai, X.; Zhou, W.; Wu, L.; Luo, J.; and Li, H. 2016. Effective active skeleton representation for low latency human action recognition. *TMM* 18(2):141–154.

Camgoz, N. C.; Hadfield, S.; Koller, O.; and Bowden, R. 2016. Using convolutional 3d neural networks for user-independent continuous gesture recognition. In *ICPR*, 49–54.

Celebi, S.; Aydin, A. S.; Temiz, T. T.; and Arici, T. 2013. Gesture recognition using skeleton data with weighted dynamic time warping. In *VISAPP*, 620–625.

Chai, X.; Wang, H.; Yin, F.; and Chen, X. 2015. Communication tool for the hard of hearings: A large vocabulary sign language recognition system. In *ACII*, 781–783.

Cui, R.; Liu, H.; and Zhang, C. 2017. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In *CVPR*, 7361–7369.

Guo, D.; Zhou, W.; Wang, M.; and Li, H. 2016. Sign language recognition based on adaptive hmms with data augmentation. In *ICIP*, 2876–2880.

Guo, D.; Zhou, W.; li, H.; and Wang, M. 2017. Online early-late fusion based on adaptive hmm for sign language recognition. *accepted to TOMM*.

Hernández-Vela, A.; Bautista, M. A.; Perez-Sala, X.; Ponce, V.; Baró, X.; Pujol, O.; Angulo, C.; and Escalera, S. 2012. Bovdw: Bag-of-visual-and-depth-words for gesture recognition. In *ICPR*, 449–452.

Huang, J.; Zhou, W.; Li, H.; and Li, W. 2015. Sign language recognition using 3d convolutional neural networks. In *ICME*, 1–6.

Ishihara, T., and Otsu, N. 2004. Gesture recognition using auto-regressive coefficients of higher-order local autocorrelation features. In *ICAFGR*, 583–588.

Koller, O.; Zargaran, S.; and Ney, H. 2017. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent cnn-hmms. In *CVPR*, 4297–4305.

Lefebvre, G.; Berlemont, S.; Mamalet, F.; and Garcia, C. 2015. Inertial gesture recognition with blstm-rnn. In *Artificial Neural Networks: Methods and Applications in Bio/Neuroinformatics*. Springer. 393–410.

Lin, Y.; Chai, X.; Zhou, Y.; and Chen, X. 2014. Curve matching from the view of manifold for sign language recognition. In *ACCV*, 233–246.

Liu, T.; Zhou, W.; and Li, H. 2016. Sign language recognition with long short-term memory. In *ICIP*, 2871–2875.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, 1412–1421.

Molchanov, P.; Yang, X.; Gupta, S.; Kim, K.; Tyree, S.; and Kautz, J. 2016. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *CVPR*, 4207–4215.

Neverova, N.; Wolf, C.; Taylor, G.; and Nebout, F. 2016. Moddrop: Adaptive multi-modal gesture recognition. *TPAMI* 38(8):1692–1706.

Pan, Y.; Mei, T.; Yao, T.; Li, H.; and Rui, Y. 2016. Jointly modeling embedding and translation to bridge video and language. In *CVPR*, 4594–4602.

Pigou, L.; Van Den Oord, A.; Dieleman, S.; Van Herreweghe, M.; and Dambre, J. 2015. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *International Journal of Computer Vision* 1–10.

Pu, J.; Zhou, W.; and Li, H. 2016. Sign language recognition with multi-modal features. In *PCM*, 252–261.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Tewari, Y. C.; Koduru, K.; Mishra, V.; and Upadhyay, P. K. 2015. American sign language recognition using haar type classifier. *KES* 19(1):63–68.

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 4489–4497.

Venugopalan, S.; Rohrbach, M.; Donahue, J.; Mooney, R.; Darrell, T.; and Saenko, K. 2015. Sequence to sequence-video to text. In *ICCV*, 4534–4542.

Wang, S. B.; Quattoni, A.; Morency, L.-P.; Demirdjian, D.; and Darrell, T. 2006. Hidden conditional random fields for gesture recognition. In *CVPR*, 1521–1527.

Wang, J.; Liu, Z.; Chorowski, J.; Chen, Z.; and Wu, Y. 2012. Robust 3d action recognition with random occupancy patterns. In *ECCV*, 872–885.

Wang, H.; Chai, X.; Zhou, Y.; and Chen, X. 2015. Fast sign language recognition benefited from low rank approximation. In *FG*, 1–6.

Wang, H.; Chai, X.; Hong, X.; Zhao, G.; and Chen, X. 2016a. Isolated sign language recognition with grassmann covariance matrices. *ACM TACCESS* 8(4):14.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016b. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 20–36.

Wu, D.; Pigou, L.; Kindermans, P.-J.; Nam, L.; Shao, L.; Dambre, J.; and Odobez, J.-M. 2016. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *TPAMI* 38(8):1583–1597.

Yang, R., and Sarkar, S. 2006. Gesture recognition using hidden markov models from fragmented observations. In *CVPR*, 766–773.

Yao, L.; Torabi, A.; Cho, K.; Ballas, N.; Pal, C.; Larochelle, H.; and Courville, A. 2015. Describing videos by exploiting temporal structure. In *ICCV*, 4507–4515.

Zhu, W.; Hu, J.; Sun, G.; Cao, X.; and Qiao, Y. 2016. A key volume mining deep framework for action recognition. In *CVPR*, 1991–1999.