TCNet: Continuous Sign Language Recognition from Trajectories and Correlated Regions

Hui Lu, Ronald Poppe, Albert Ali Salah [1*]

[1]Utrecht University

Abstract

A key challenge in continuous sign language recognition (CSLR) is to efficiently capture long-range spatial interactions over time from the video input. To address this challenge, we propose TCNet, a hybrid network that effectively models spatio-temporal information from **T**rajectories and **C**orrelated regions. TCNet's trajectory module transforms frames into aligned trajectories composed of continuous visual tokens. In addition, for a query token, self-attention is learned along the trajectory. As such, our network can also focus on fine-grained spatio-temporal patterns, such as finger movements, of a specific region in motion. TCNet's correlation module uses a novel dynamic attention mechanism that filters out irrelevant frame regions. Additionally, it assigns dynamic key-value tokens from correlated regions to each query. Both innovations significantly reduce the computation cost and memory. We perform experiments on four large-scale datasets: PHOENIX14, PHOENIX14-T, CSL, and CSL-Daily, respectively. Our results demonstrate that TCNet consistently achieves state-of-the-art performance. For example, we improve over the previous state-of-the-art by 1.5% and 1.0% word error rate on PHOENIX14 and PHOENIX14-T, respectively.

Introduction

Continuous sign language recognition (CSLR) is the task of transcribing sequences of continuous gestures made by a signer into sentences. Unlike isolated sign language recognition, CSLR algorithms are typically trained in a weakly-supervised way because only sentence-level labels are provided. Temporal information is only available implicitly. One challenge is to extract relevant patterns over time.

Traditionally, CSLR has been addressed by first extracting spatial patterns from video, and subsequently considering temporal relations between these features, typically using LSTMs (e.g., Min et al. (2021); Niu and Mak (2020)). Since sign language is mainly understood from signers' faces and hands, some CSLR models (Papadimitriou and Potamianos 2020; Zhou et al. 2020) leverage pre-trained pose detectors to locate the face and hands and then crop the feature maps to form a multi-stream architecture. This approach allows to focus on the potentially relevant regions in the video input, but is unsuited to exploit interactions between these regions.

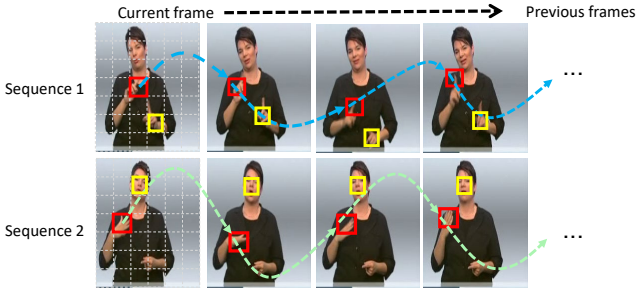

Figure 1: **Trajectories and correlated regions**. The sign is revealed by the trajectories of both hands in Seq. 1, and the right hand and head in Seq. 2. The content of these regions, such as the hand pose, is also important. We visualize the trajectory of the right hand (red) and indicate correlated regions (yellow).

More recently, transformer models allow focusing on informative spatial regions in the input by employing self-attention (Hu et al. 2023a,b). Although self-attention has been shown to improve recognition performance, its high computational cost limits its use to narrow spatial windows. Moreover, required pairwise token affinity computations across all spatial locations amount to a high computational complexity and incur significant memory footprints.

For the CSLR task, correlated regions are generally dynamic. In Figure 1, the hand regions are correlated in Seq.1, but not in Seq.2. This requires us to assign dynamic key-value tokens from correlated regions to each query, such that irrelevant tokens can be avoided.

We build on these notions by explicitly focusing on extracting trajectories, considering patterns along these trajectories, and modeling the correlations between relevant regions, without making any assumptions about these regions. We propose a novel network named TCNet, which collects temporal information over an entire sequence and attends to correlated regions to collect spatial information.

Our main innovations are the trajectory and correlation modules. The trajectory module transforms temporal move-

*This is the uncorrected author proof, including the supplementary material. Please cite as: H. Lu, R. Poppe, A.A. Salah, "TCNet: Continuous Sign Language Recognition from Trajectories and Correlated Regions", Proc. AAAI, Vancouver, 2024.

ments into pre-aligned trajectories of visual tokens, without relying on face or hand detection. Additionally, self-attention is calculated along these trajectories. The correlation module calculates attention per frame, where only the correlated regions are considered. TCNet can effectively extract spatial-temporal information with significantly reduced computation cost and memory compared to related work, while showing improved performance.

In this paper, we make the following contributions:

- We propose TCNet, a hybrid network for CSLR that employs innovative trajectory and correlation modules in the feature extractor to improve the feature representation.
- Both modules are combined to yield an effective network with reduced computation and memory cost.
- TCNet achieves state-of-the-art results on PHOENIX14, PHOENIX14-T, CSL, and CSL-Daily. Our ablation studies demonstrate the added value of trajectory and correlation modules. Experiments on various backbones demonstrate consistent performance gains.
- We use standard experimental protocols and release all our code for reproducibility.

Related Work

The goal of CSLR is to translate sequences of image frames into corresponding glosses, the atomic lexical units of sign language. The task is weakly supervised because only sentence-level labels are provided. Hence, there is a challenge in extracting relevant spatio-temporal patterns from the input frames. Recent progress in CSLR is mainly due to advances in deep learning. We discuss current work in extracting spatial and temporal patterns, respectively.

Extracting spatial patterns

Several methods (Min et al. 2021; Niu and Mak 2020) employ 2D CNN models to extract frame-wise features, and adopt LSTMs for long-term temporal modeling. Since the most informative regions in the video input correspond to the face and hands (Koller 2020), some CSLR models (Papadimitriou and Potamianos 2020; Zhou et al. 2020, 2021b) leverage pre-trained face/hand detectors or pose estimation algorithms (e.g., Cao et al. (2021); Sun et al. (2019)) to localize relevant parts of the body. The corresponding regions are cropped from the input and used as additional inputs in multi-stream architectures. This strategy allows to focus on relevant parts of the input but interactions between the regions cannot be readily used (Zuo and Mak 2022).

Inspired by recent progress in transformers, researchers have used self-attention for CSLR to enhance the feature extractor, by focusing on relations between regions within a single frame (Hu et al. 2023a,b). Although self-attention has been shown to improve recognition performance since the receptive field is increased, the high computational cost restricts it use to limited spatial windows. Moreover, there is a significant computation cost involved in calculating the pairwise token affinity between all potential spatial locations.

To address this issue, researchers have proposed attention variants to extract correlated regions and to reduce the computation cost. For example, Zuo and Mak (2022) use a lightweight spatial attention module guided by keypoint heatmaps to enforce a focus on informative regions.

We argue that, specifically for the CSLR task, not all regions in the frame contribute equally to recognition since the sign language are mainly conveyed through head and hand regions, Moreover, the correlated regions are generally dynamic. There is a need for mechanisms to attend to regions that can dynamically vary over time.

Exploiting temporal information

Recently, there is an increased focus on extracting temporal information (Hu et al. 2023a,b). By leveraging the cross-frame temporal movements to express a sign, temporal patterns, for example of the hand and face, can be learned (Hu et al. 2023a). Current approaches mainly use 3D convolutions (Pu, Zhou, and Li 2019) or their (2+1)D variants (Tran et al. 2018) to capture short-term temporal information.

More flexible ways of processing the temporal information, such as Temporal shift (Lin, Gan, and Han 2019) or temporal convolutions (Liu et al. 2020), also focus on short-term temporal movements. Recently, researchers have used adjacent frames to extract temporal information. Hu et al. (2023b) extract temporal features from the difference between adjacent frames as approximate motion information, and then concatenate it with appearance features as input to self-attention, such that the network can capture spatio-temporal information. Similarly, CorrNet (Hu et al. 2023a) uses a self-attention mechanism in temporally adjacent frames to collect body movement features. However, these approaches are limited to adjacent frames thus can only capture short-term temporal information.

Relevant movements can be temporally more or less distant. However, aggregating distinctive information from distant regions remains a challenge due to the limited spatial-temporal receptive field. To model variations, Guo et al. (2023) propose a dual-pathway network where two branches accommodate the local and global temporal context, respectively. We deviate from this approach by elegantly attending to dynamically varying regions, as are typically in sign language due to the continuous movement of the hands.

Method

We introduce TCNet, a hybrid CNN-attention network that is capable of focusing on dynamically changing informative regions. To this end, we propose TCNet blocks that combine novel trajectory and correlation modules to enhance the spatio-temporal feature extraction capabilities.

Overall architecture

The architecture of TCNet is shown in Figure 2. TCNet is based on previous works, including TLP (Hu et al. 2022) and CorrNet (Hu et al. 2023a), as we use the same feature extraction backbone, sequential modeling and classifier. This allows us to make pairwise comparisons to demonstrate the merits of the novel modules in the feature extraction module. But the TCNet block can be incorporated in a range of backbones, as we demonstrate in the ablation studies.

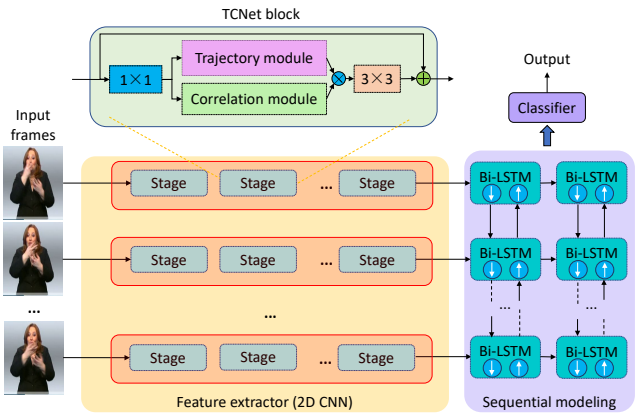

Figure 2: **TCNet architecture** with feature extractor, sequential modeling and classifier. TCNet blocks with our trajectory and correlation modules extract spatio-temporal features at various sequential stages.

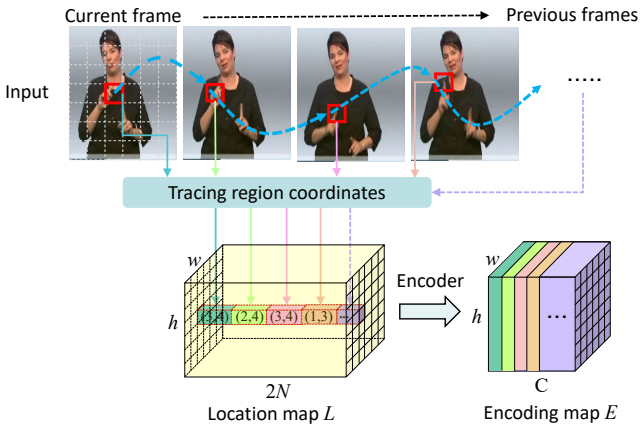

Figure 3: **Calculation of location map L and encoding map E** in the trajectory module. For each frame, regions are traced back. Coordinates of a region in previous frames are stored at the same location. The location map is passes an encoder to provide encoding map E.

Continuous sign language recognition deals with a sequence mapping from a sign language video with T frames $x = \{x_t \in \mathbb{R}^{w \times h \times c}\} = \{x_t\}_{t=1}^{T}$ to a sequence $y = \{y_i\}_{i=1}^{L}$ of length L, where $w \times h$ is the size of an input frame x_t, and the number of channels c is three for RGB inputs. Our CSLR model consists of a feature extractor, a sequential modeling module and a classifier (see Figure 2). The feature extractor first processes input frames into fixed-length frame-wise features, $v = \{v_t\}_{t=1}^{T} \in \mathbb{R}^{T \times d}$, with d the number of neurons in the last fully connected layer of the backbone. In case of a ResNet-18 backbone, $d = 512$. The processing is sequential and follows several stages, in line with previous work (Hu et al. 2022, 2023a). Details of the network architecture appear in the supplementary material. After feature extraction, a series of 2-layer Bi-directional Long Short-Term Memory (Bi-LSTMs) perform temporal modeling based on these extracted visual representations. Finally, a classifier provides the gloss predictions.

TCNet block. We introduce a TCNet block that is responsible for more effective extraction of informative spatio-temporal features. The block operates on subsequences of N frames ($N < T$). The trajectory and correlation modules are processed in parallel (see top of Figure 2), after which the feature maps from both modules are aggregated through element-wise multiplication.

Trajectory module

The trajectory module is motivated by our desire to both capture the trajectories of moving regions, and to apply self-attention to those regions under movement. To this end, we leverage a light-weight motion estimation algorithm (Ranjan and Black 2017) that operates on pairs of input frames to construct an initial location map, which is then encoded. This map provides distinctive information about the movement of the regions. Finally, we obtain the temporal attention map by applying self-attention along the trajectories. The process is illustrated in Figure 3.

Location map. The location map L will trace back coordinates of regions as the trajectory, which is represented as a group of matrices over time. For an input subsequence of N frames, we obtain $N-1$ backwards optical flow maps l_t ($1 < t \leq N$) by considering pairs of subsequent frames. l_t is a volume of size $w \times h \times 2$, with $w \times h$ the frame resolution and two channels for the x- and y-coordinates. We process the pair of frames in such a way that a coordinate in the optical flow map points to the coordinate in the earlier frame. Specifically, we reverse the order of the pair of frames, and round the floating point optical flow estimates to the nearest coordinates. Each element in l_t thus corresponds to a location in frame $t-1$. Finally, l_1 corresponds to the location indices, typically the pixel coordinates.

We then build the location map $L \in \mathbb{R}^{w \times h \times 2N}$ from all l_t. We recursively trace back the location of the region in the previous frame such that each coordinate in L represents the location of the current region in the previous frame. Specially, the coordinate of the same region in previous frames will be recorded at the same location of current frame.

For example, in Figure 3, the coordinate of the red region in current frame is (3,4), and the coordinates in the previous three frames are (2,4), (3,4) and (1,3). These coordinate will be written in the same location of the location map (3,4). The vector at each coordinate, through the channel dimension, can be considered a trajectory with alternating x- and y-coordinates over time.

The location map is subsequently encoded through two 1×1 convolutions to reduce the depth and outputs. The output is encoding map E with the dimensions $w \times h \times C$, with C depending on the stage of the feature extractor. The architecture of the network is provided in the supplementary material. In our design, the trajectory generation can be expressed as efficient matrix operations.

Temporal attention map. We first populate a temporally pre-aligned map by changing the coordinates in L by the ac-

tual values of the corresponding time frame. In this process, we effectively cancel out gross movement, which allows us to focus on finer-grained movement such as hand signs. For the query in the current frame, we select the keys and values along the trajectories and calculate self-attention to get the feature map related to the trajectories. The calculation of the temporal self-attention volume TA over the input subsequence follows the multi-head scaled dot-product attention (Vaswani et al. 2017). The generated temporal attention map TA is then added to encoding map E, and passed through a 1×1 convolution as the final result.

Correlation module

To extract the spatial information, several works (Hu et al. 2023a,b) use dilated convolution or self-attention on adjacent frames to emphasize relevant regions. To reduce the computation cost and to reduce the blur introduced from calculating the weighted sum including irrelevant tokens, several works (e.g., Dong et al. (2022); Xia et al. (2022); Zhu et al. (2023)) propose different sparse attention mechanisms such as local windows (Dong et al. 2022), deformable windows (Xia et al. 2022), or use a small sampled subset of key-value pairs for the query (Zhu et al. 2023)).

However, in the CSLR task, we argue that not all regions in the frame contribute equally to the recognition, since sign language is mainly conveyed through the head and hand regions. Moreover, the correlated regions are dynamic. Hence, it will be more effective if we can design dynamic key-value tokens from correlated regions for each query when calculating self-attention. Based on these observations, we propose a correlation module with a dynamic attention mechanism. Our design significantly reduces the computation cost and memory use.

Our key idea is to dynamically filter out irrelevant key-value pairs based on the input feature map. To simplify the notations, we discuss the case of single-head self-attention with a single input, although we use multi-head self-attention in practice. The process is shown in Figure 4. Given a 2D input feature map $X \in \mathbb{R}^{w \times h \times c}$, we first split it into non-overlapped regions using window size K. Each region $X^r \in \mathbb{R}^{K \times K \times c}$ and the corresponding query, key, and value are obtained with linear projections (1×1 convolution):

$$Q^r = X^r W^q, K^r = X^r W^k, V^r = X^r W^v \quad (1)$$

where $W^q, W^k, W^v \in \mathbb{R}^{1 \times 1 \times c}$ are projection weights for the query, key and value, respectively.

For region-level queries Q^r and keys K^r, the region-to-region affinity matrix, A^r, can be obtained via matrix multiplication between Q^r and the transposed K^r:

$$A^r = Q^r (K^r)^T \quad (2)$$

An example of calculating the regional affinity matrix A^{R1} for Region 1 is shown in Figure 4. The value of every entry in affinity matrix A^r measures how much two tokens are semantically related. For example, in Figure 4, the first line indicates the affinity of token $t1$ to tokens from key1, key2, key3, and key4. To reduce the computation cost and

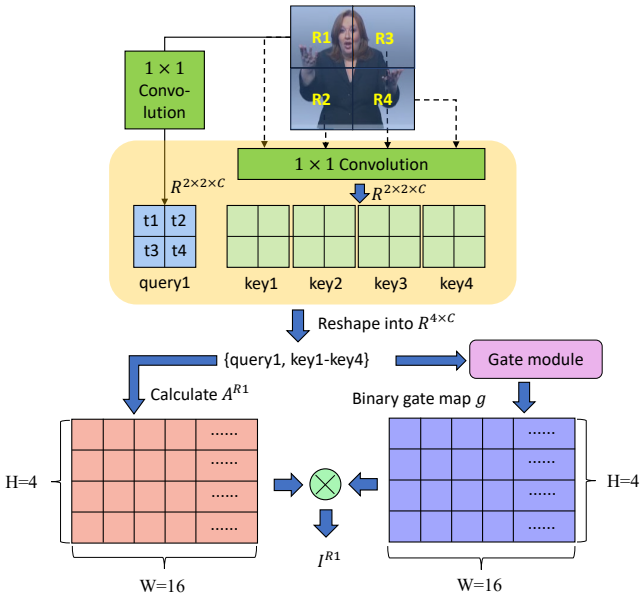

Figure 4: **Calculation of the sparse attention matrix** I^r in the correlation module. In this example, our input is a 4×4 image and $K = 2$. The frame is split into four regions R1-R4, which pass through a 1×1 convolution to yield queries and keys. For R1 (other regions analogous), the left branch provides affinity matrix A^{R1}, while the right branch generates binary gate map g. Both volumes are multiplied element-wisely and pruned to produce sparse attention matrix I^{R1}. By combining these results for all regions, we get sparse and dynamic affinity matrix I^R for the current frame.

to reduce blur introduced by having irrelevant tokens contribute to the weighted sum, the core step that we perform next is to prune the affinity graph by dynamically keeping k connections for each region. To realize this, we use a gate module to generate a binary gate map g. Through element-wise multiplication with A^r, we obtain a sparse matrix I^r. Hence, the i^{th} row of I^r contains k indices of the most relevant regions for the i^{th} region. Since the generated g depends on the input frames, the output g is fully dynamic.

For a region with $\{query1, key1, key2, key3, key4\} \in \mathbb{R}^{K^2 \times c}$, the input of the gate module is calculated as the concatenation of the vector product of $query1$ with each of the four transposed keys.

The gate module consists of two 3×3 convolutions with a padding and stride of one. The module outputs an initial gate map g' that has the same size as A^r. We generate binary gate map g from g' by pruning the affinity graph. To make sure we can back-propagate the error through the discrete g, we adopt Improved SemHash (Kaiser and Bengio 2018). We define the function to get the binary gate feature map g as follows:

$$g_\alpha = \sigma'(g') \quad and \quad g_\beta = 1(g' > 0) \quad (3)$$

where $\sigma'(x)$ is the saturating sigmoid function (Kaiser and

Bengio 2016; Kaiser and Sutskever 2015):

$$\sigma'(x) = max(0, min(1, 1.2\sigma(x) - 0.1)) \tag{4}$$

with sigmoid function σ.

Here, g_α is a real-valued gate map with all entries falling in the interval $[0.0, 1.0]$, while g_β is a binary gate map. We observe that g_β has the desirable binary property that we want to use in our model, but the gradient of g_β w.r.t g' is not defined. On the other hand, the gradient of g_α w.r.t g' is well defined, but g_α is not binary. In forward propagation, we randomly use $g = g_\alpha$ for half of the training samples and use $g = g_\beta$ for the remaining samples. When g_β is used, we follow the solution in (Kaiser and Bengio 2018) and define the gradient of g_β w.r.t g' to be the same as the gradient of g_α w.r.t g' in the backward propagation. In the evaluation and inference phases, we use $g = g_\beta$.

Classifier

The final classifier is essentially a fully connected layer. It performs classification based on the temporal features extracted by the Bi-LSTM layer. The output of the Bi-LSTM layer is a 2D tensor with dimensions (batch size, hidden size), where hidden size is the size of the LSTM hidden states. In practice, we initialize the classifier with Xavier Uniform initialization. The output of the Bi-LSTM layer is multiplied with the weight matrix to generate the final output.

Experiments

We provide extensive experiments to evaluate the effectiveness of our method. Datasets and evaluation metrics are introduced first. Then, we detail the experimental setup and analyze the results.

Dataset and metrics

We conduct our experiments on four public datasets.

PHOENIX14 (Koller, Forster, and Ney 2015) is recorded from a German weather forecast broadcast with nine actors before a clean background with a resolution of 210×260. It contains 6,841 sentences with a vocabulary of 1,295 signs, divided into 5,672 training, 540 development (DEV), and 629 testing (TEST) samples.

PHOENIX14-T (Camgoz et al. 2018) covers the same domain as PHOENIX14 and is available for both CSLR and sign language translation tasks. It has a vocabulary of 1,085 signs and contains 8,247 sentences. These are divided over training (7,096 instances), development (DEV, 519 samples), and testing (TEST, 642 samples) sets.

CSL (Huang et al. 2018) is collected in the laboratory environment by fifty signers with a vocabulary size of 178 with 100 sentences. It contains 25,000 videos, divided into training and testing sets by a ratio of 4:1.

CSL-Daily (Zhou et al. 2021a) revolves the daily life, recorded indoors at 30fps by 10 signers. CSL-Daily has the largest vocabulary size (2k) among the tested datasets, covering a broad range of topics. It contains 20,654 sentences, divided into 18,401 training samples, 1,077 development (DEV) samples and 1,176 samples for testing (TEST).

In continuous SLR, word error rate (WER) is the most widely-used metric to evaluate the performance. WER is essentially an edit distance and reflects the lowest number of **sub**stitution, **ins**ertion, and **del**etion operations to transform the predicted sentence into the reference sequence:

$$\text{WER} = \frac{\text{\#sub + \#ins + \#del}}{\text{length of reference}} \tag{5}$$

Training details

Following previous works, ours uses a ResNet-18 backbone. We train the network using the following loss function:

$$\mathcal{L} = \mathcal{L}_{CTC} + \mathcal{L}_{VE} + \mathcal{L}_{VA} + \frac{||G(X)||_1}{c} \tag{6}$$

where $\mathcal{L}_{CTC}$ is the popular CTC loss (Graves et al. 2006) to predict the probability of the target gloss, and $\mathcal{L}_{VE}$ and $\mathcal{L}_{VA}$ are VE and VA losses from VAC (Min et al. 2021). The final term is a L1 regularization term to encourage g to be sparse. Here, c is the size of g. The backbone network receives no gradients from the second term, while the gate module receives gradients from the first two terms in Equation 6.

In the trajectory module, we use the pre-trained SPyNet (Ranjan and Black 2017) as motion estimation network and we set size of the subsequences to $N = 12$. An ablation study is performed later. The number of hidden states in the 2-layer Bi-LSTM encoder is set to be 1,024, followed by a fully connected layer for gloss prediction. We train our model for 80 epochs using the Adam optimizer. The initial learning rate is set to be 1×10^{-4} decayed by 5 after 40 and 60 epochs, and the weight decay is set to 5×10^{-3}. All input frames are first resized to 256×256, and then randomly cropped to 224×224 with 50% horizontal flipping and 20% temporal rescaling during training. During inference, a 224×224 center crop is used.

Comparison with state-of-the-art

We first compare TCNet to the current state-of-the-art.

PHOENIX14 and PHOENIX14-T. A comparison between TCNet and current state-of-the-art methods on PHOENIX14 and PHOENIX14-T appears in Table 1. We split the table into three parts. The middle part, works annotated with $*$, contains results that are obtained using additional inputs like face or hand features.

TCNet outperforms state-of-the-art methods by a clear margin on both datasets. Compared to methods without additional inputs, TCNet obtains higher performance than the second best CorrNet on both PHOENIX14 (18.8→18.1 and 19.4→18.9 on DEV and TEST, respectively) and PHOENIX14 (18.9→18.3 and 20.5→19.4 on DEV and TEST). Compared to methods with additional inputs, our TCNet still achieves better performance, since the advanced design of the correlation module can dynamically extract correlated information from face or hand regions, and the trajectory module can trace the trajectories of these regions. For example, compared to C^2SLR, with the same backbone, TCNet lowers the WER by 2.4% and 1.5% on PHOENIX14 DEV and TEST, respectively (20.5→18.1, 20.4→18.9). Although C^2SLR outperforms CorrNet by

Method	Backbone	FLOPs	Params	PHOENIX14				PHOENIX14-T	
				DEV		TEST		DEV	TEST
				del / ins	WER	**del / ins**	WER	WER	WER
SFL (Niu and Mak 2020)	ResNet18	1180.2G	33.7M	7.9 / 6.5	26.2	7.5 / 6.3	26.8	25.1	26.1
FCN (Cheng et al. 2020)	Custom	983.0G	20.3M	7.8 / 4.7	23.7	7.2 / 4.5	23.9	23.3	25.1
CMA (Pu et al. 2020)	GoogLeNet	-	-	7.3 / 2.7	21.3	7.3 / 2.4	21.9	-	-
VAC (Min et al. 2021)	ResNet18	1120.4G	22.3M	7.9 / 2.5	21.2	8.4 / 2.6	22.3	21.4	23.9
SMKD (Hao, Min, and Chen 2021)	ResNet18	1032.3G	20.3M	6.8 / 2.5	20.8	6.3 / 2.3	21.0	20.8	22.4
TLP (Hu et al. 2022)	ResNet18	2950.5G	48.0M	6.3 / 2.8	19.7	6.1 / 2.9	20.8	19.4	21.2
SEN (Hu et al. 2023b)	ResNet18	1144.2G	23.1M	5.8 / 2.6	19.5	7.3 / 4.0	21.0	19.3	20.7
CorrNet (Hu et al. 2023a)	ResNet18	1035.4G	20.5M	5.6 / 2.8	18.8	5.7 / 2.3	19.4	18.9	20.5
SLT* (Camgoz et al. 2018)	GoogLeNet	-	-	-	-	-	-	24.5	24.6
CNN+LSTM+HMM* (Koller et al. 2019)	GoogLeNet	1488.2G	44.0M	8.1 / 3.8	26.0	8.2 / 3.6	26.0	22.1	24.1
DNF* (Cui, Liu, and Zhang 2019)	GoogLeNet	1487.3G	43.30M	7.3 / 3.3	23.1	6.7 / 3.3	22.9	22.7	24.0
STMC* (Zhou et al. 2020)	VGG11	1742.1G	40.71M	7.7 / 3.4	21.1	7.4 / 2.6	20.7	19.6	21.0
C^2SLR* (Zuo and Mak 2022)	ResNet18	1124.6G	32.6M	6.8 / 3.0	20.5	7.1 / 2.5	20.4	20.2	20.4
TCNet (ours)	ResNet18	932.1G	19.3M	**5.5 / 2.4**	**18.1**	**5.4 / 2.0**	**18.9**	**18.3**	**19.4**

Table 1: Comparison with state-of-the-art on PHOENIX14 and PHOENIX14-T. WER in %. Networks with $*$ are trained using additional inputs such as face or hand features, or pre-extracted heatmaps. Best results in **bold**.

0.1% on PHOENIX14-T TEST, TCNet surpasses C^2SLR by a 1.0%(20.4→19.4) WER reduction. We see the same pattern for the number of deletions and insertions.

TCNet has the lowest number of FLOPs and trained parameters of all the tested networks. These include the additional operations and parameters of the motion estimation module. The improved performance therefore is not due to the increased model capacity.

CSL. Table 8 shows that our TCNet achieves the new state-of-the-art performance, lowering the previous WER of 0.8 (Hu et al. 2023a,b) to 0.7. Again, TCNet has the lowest number of FLOPs and parameters from the tested methods.

Method	FLOPs	Params	WER (%)
SF-Net (Yang et al. 2019)	1056.1G	27.9M	3.8
FCN (Cheng et al. 2020)	977.2G	20.3M	3.0
STMC (Zhou et al. 2020)	1726.1G	40.71M	2.1
VAC (Min et al. 2021)	1107.4G	22.3M	1.6
C^2SLR (Zuo and Mak 2022)	1107.2G	31.7M	0.9
CorrNet (Hu et al. 2023a)	1033.1G	20.3M	0.8
SEN (Hu et al. 2023b)	1136.1G	23.0M	0.8
TCNet (ours)	929.8G	18.8M	**0.7**

Table 2: Comparison with state-of-the-art methods on CSL. Best results in **bold**.

CSL-Daily. Table 3 shows that TCNet also outperforms all state-of-the-art works upon on this challenging dataset. In particular, we reduce the previous best TEST WER of CorrNet (Hu et al. 2023a) by 0.8%. These results demonstrate the merits of TCNet to address a more general input domain.

Ablation study

To analyze the merits of the TCNet block, we perform ablations. First, we assess the contribution of the trajectory and correlation modules individually on the performance. Second, we investigate the influence of the backbone. Third, we experiment with various combination operations to combine results from the two modules. Fourth, we show the performance with input subsequences of various lengths N.

Method	DEV WER	TEST WER
LS-HAN (Huang et al. 2018)	39.0	39.4
TIN-Iterative (Cui, Liu, and Zhang 2019)	32.8	32.4
Joint-SLRT (Camgoz et al. 2020)	33.1	32.0
FCN (Cheng et al. 2020)	33.2	32.5
BN-TIN (Zhou et al. 2021a)	33.6	33.1
SEN (Hu et al. 2023b)	31.1	30.7
CorrNet (Hu et al. 2023a)	30.6	30.1
TCNet (ours)	**29.7**	**29.3**

Table 3: Comparison with state-of-the-art methods on CSL-Daily. WER in %. Best results in **bold**.

Influence of trajectory and correlation modules. We assess the contribution of the two modules separately using the same experimental setting on PHOENIX14. The results are summarized in Table 4. Compared with the baseline, the trajectory module and correlation module each bring a notable reduction in WER in the range 1.0-1.6%. When both modules are used together, there is a further reduction of 0.5-1.1%. This shows that the two modules are both effective, and their influence is partly complementary.

Trajectory	Correlation	DEV WER (%)	TEST WER (%)
		20.2	21.0
	✓	18.8 (-1.4)	19.8 (-1.2)
✓		19.2 (-1.0)	19.4 (-1.6)
✓	✓	**18.1** (-2.1)	**18.9** (-2.1)

Table 4: Influence of the trajectory and correlation modules on PHOENIX14. Best results in **bold**.

Influence of backbone. Instead of the ResNet-18, we incorporate TCNet blocks in backbones GoogLeNet (Szegedy

et al. 2015), VGG11 (Simonyan and Zisserman 2014), SqueezeNet (Hu, Shen, and Sun 2018), and ShuffleNet V2 (Ma et al. 2018). The aggregated result of the trajectory and correlation modules are added with three spatial downsampling layers in each backbone. Using the same experimental procedure as before, we summarize the results on PHOENIX14 in Table 5. We observe that the use of TCNet blocks consistently improves the performance of the backbones with a reduction in WER in the range 1.8-2.6%. These results demonstrate the choice of backbone matters, but the inclusion of the trajectory and correlation modules is always beneficial.

Backbone	DEV WER	TEST WER
SqueezeNet (Hu, Shen, and Sun 2018)	22.2	22.6
+TCNet	**19.8** (-2.4)	**20.0** (-2.6)
ShuffleNet V2 (Ma et al. 2018)	21.7	22.2
+TCNet	**19.4** (-2.3)	**20.0** (-2.2)
GoogLeNet (Szegedy et al. 2015)	21.4	21.5
+TCNet	**19.0** (-2.4)	**19.3** (-2.2)
VGG11 (Simonyan and Zisserman 2014)	20.7	21.0
+TCNet	**18.7** (-2.0)	**19.2** (-1.8)

Table 5: Backbones with and without TCNet blocks on PHOENIX14. WER in %. Best results in **bold**.

Combination of trajectory and correlation modules. We experiment with alternative operators to aggregate the results from the trajectory and correlation modules: summation and concatenation. Comparisons with the element-wise multiplication are presented in Table 6. The difference in performance is modest, with only 0.1-0.4% higher WER for alternative ways of combining the module outputs.

Aggregation operation	DEV WER (%)	TEST WER (%)
Concatenation	18.5	19.2
Summation	18.3	19.0
Element-wise multiply	**18.1**	**18.9**

Table 6: Alternative combination operators of trajectory and correlation modules on PHOENIX14. Best results in **bold**.

Number of frames for tracing the trajectory. In the trajectory module, we trace the trajectories in subsequences of length N. Since the number of number may affect the effectiveness of the temporal information encoding, we investigate the performance on PHOENIX14 for various N. We summarize the results in Table 7. The best performance is achieved for $N = 12$, approximately half a second, our default setting. Increasing or decreasing the number of frames gradually increases the WER.

Frames	1	2	4	8	12	16	18
DEV	18.8	18.7	18.3	18.2	18.1	18.9	19.3
TEST	19.5	19.5	19.2	19.0	18.9	19.5	20.0

Table 7: Subsequence length N in the trajectory module on PHOENIX14. WER in %. Best results in **bold**.

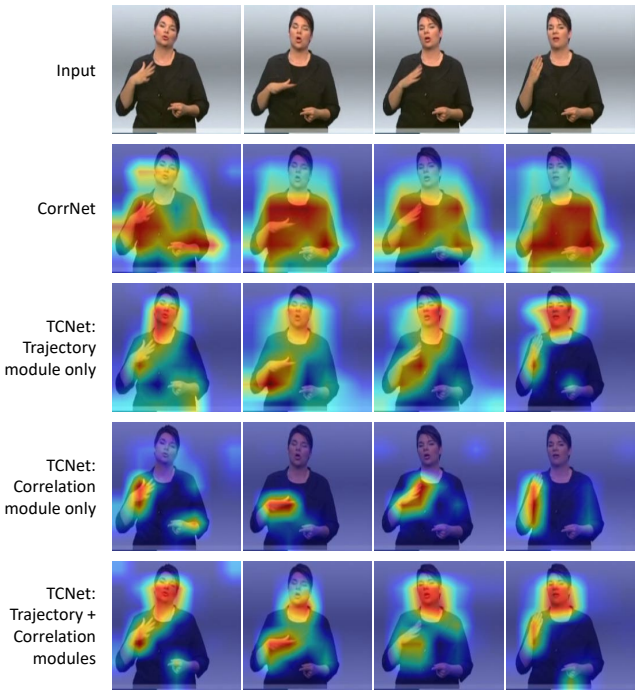

Figure 5: Grad-CAM heatmaps for CorrNet and TCNet with only the Trajectory module, only the Correlation module, and with both modules.

Visualizations

In Figure 5, we show Grad-CAM (Selvaraju et al. 2017) heatmaps for CorrNet (Hu et al. 2023a) and TCNet with either and both trajectory and correlation module in the TCNet block. While CorrNet is relatively unfocused and also attends to irrelevant regions, TCNet focuses almost exclusively on the face and right hand that performs the gestures.

The trajectory module mainly focuses on the head and right hand regions of the signer since the body movement mainly happens in these regions, while the left hand remains almost stationary. Overall, the attention is not too narrow. In contrast, the correlation module focuses mostly on the right hand, a bit on the left hand but not on the face. The two modules combined demonstrate attention for the most informative regions to understand the sign.

Conclusion

In this work, we have addressed the extraction of more informative spatio-temporal features for continuous sign language recognition (CSLR). We have proposed novel trajectory and correlation modules that are incorporated in a hybrid network: TCNet. The trajectory module traces movements of regions, mainly of the hands and face, over time to allow for self-attention along the trajectory. The correlation module dynamically filters out irrelevant key-value pairs. Based on the aggregation of both modules, we have demonstrated state-of-the-art results on common CSLR datasets. For example, we improve over the previous state-of-the-art (Hu et al. 2023a) by 1.5% and 1.0% word error rate

on PHOENIX14 and PHOENIX14-T, respectively. Ablation studies validate the contribution of both modules, and the robustness of the results with different design choices.

References

Camgoz, N. C.; Hadfield, S.; Koller, O.; Ney, H.; and Bowden, R. 2018. Neural sign language translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7784–7793.

Camgoz, N. C.; Koller, O.; Hadfield, S.; and Bowden, R. 2020. Sign language transformers: Joint end-to-end sign language recognition and translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10023–10033.

Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.-E.; and Sheikh, Y. 2021. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1): 172–186.

Cheng, K. L.; Yang, Z.; Chen, Q.; and Tai, Y.-W. 2020. Fully convolutional networks for continuous sign language recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, 697–714. Springer.

Cui, R.; Liu, H.; and Zhang, C. 2019. A deep neural framework for continuous sign language recognition by iterative training. *IEEE Transactions on Multimedia*, 21(7): 1880–1891.

Dong, X.; Bao, J.; Chen, D.; Zhang, W.; Yu, N.; Yuan, L.; Chen, D.; and Guo, B. 2022. CSWin Transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12124–12134.

Graves, A.; Fernández, S.; Gomez, F.; and Schmidhuber, J. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, 369–376.

Guo, L.; Xue, W.; Guo, Q.; Liu, B.; Zhang, K.; Yuan, T.; and Chen, S. 2023. Distilling Cross-Temporal Contexts for Continuous Sign Language Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10771–10780.

Hao, A.; Min, Y.; and Chen, X. 2021. Self-mutual distillation learning for continuous sign language recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11303–11312.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Hu, L.; Gao, L.; Liu, Z.; and Feng, W. 2022. Temporal lift pooling for continuous sign language recognition. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV*, 511–527. Springer.

Hu, L.; Gao, L.; Liu, Z.; and Feng, W. 2023a. Continuous Sign Language Recognition with Correlation Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2529–2539.

Hu, L.; Gao, L.; Liu, Z.; and Feng, W. 2023b. Self-Emphasizing Network for Continuous Sign Language Recognition. In *Thirty-seventh AAAI conference on artificial intelligence*.

Huang, J.; Zhou, W.; Zhang, Q.; Li, H.; and Li, W. 2018. Video-based sign language recognition without temporal segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2257–2264.

Kaiser, Ł.; and Bengio, S. 2016. Can active memory replace attention? *Advances in Neural Information Processing Systems*, 29.

Kaiser, Ł.; and Bengio, S. 2018. Discrete autoencoders for sequence models. *arXiv preprint arXiv:1801.09797*.

Kaiser, Ł.; and Sutskever, I. 2015. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*.

Koller, O. 2020. Quantitative survey of the state of the art in sign language recognition. *arXiv preprint arXiv:2008.09918*.

Koller, O.; Camgoz, N. C.; Ney, H.; and Bowden, R. 2019. Weakly supervised learning with multi-stream CNN-LSTM-HMMs to discover sequential parallelism in sign language videos. *IEEE transactions on pattern analysis and machine intelligence*, 42(9): 2306–2320.

Koller, O.; Forster, J.; and Ney, H. 2015. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding*, 141: 108–125.

Lin, J.; Gan, C.; and Han, S. 2019. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, 7083–7093.

Liu, Z.; Luo, D.; Wang, Y.; Wang, L.; Tai, Y.; Wang, C.; Li, J.; Huang, F.; and Lu, T. 2020. TEINet: Towards an efficient architecture for video recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11669–11676.

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, 116–131.

Min, Y.; Hao, A.; Chai, X.; and Chen, X. 2021. Visual alignment constraint for continuous sign language recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11542–11551.

Niu, Z.; and Mak, B. 2020. Stochastic fine-grained labeling of multi-state sign glosses for continuous sign language recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, 172–186. Springer.

Papadimitriou, K.; and Potamianos, G. 2020. Multimodal Sign Language Recognition via Temporal Deformable Convolutional Sequence Learning. In *Interspeech*, 2752–2756.

Pu, J.; Zhou, W.; Hu, H.; and Li, H. 2020. Boosting continuous sign language recognition via cross modality augmentation. In *Proceedings of the 28th ACM International Conference on Multimedia*, 1497–1505.

Pu, J.; Zhou, W.; and Li, H. 2019. Iterative alignment network for continuous sign language recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4165–4174.

Ranjan, A.; and Black, M. J. 2017. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4161–4170.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 618–626.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, K.; Xiao, B.; Liu, D.; and Wang, J. 2019. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5693–5703.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 6450–6459.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30.

Xia, Z.; Pan, X.; Song, S.; Li, L. E.; and Huang, G. 2022. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4794–4803.

Yang, Z.; Shi, Z.; Shen, X.; and Tai, Y.-W. 2019. Sf-net: Structured feature network for continuous sign language recognition. *arXiv preprint arXiv:1908.01341*.

Zhou, H.; Zhou, W.; Qi, W.; Pu, J.; and Li, H. 2021a. Improving sign language translation with monolingual data by sign back-translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1316–1325.

Zhou, H.; Zhou, W.; Zhou, Y.; and Li, H. 2020. Spatial-temporal multi-cue network for continuous sign language recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 13009–13016.

Zhou, H.; Zhou, W.; Zhou, Y.; and Li, H. 2021b. Spatial-temporal multi-cue network for sign language recognition and translation. *IEEE Transactions on Multimedia*, 24: 768–779.

Zhu, L.; Wang, X.; Ke, Z.; Zhang, W.; and Lau, R. 2023. BiFormer: Vision Transformer with Bi-Level Routing Attention. *arXiv preprint arXiv:2303.08810*.

Zuo, R.; and Mak, B. 2022. C2SLR: Consistency-enhanced continuous sign language recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5131–5140.

Supplementary Material - TCNet: Continuous Sign Language Recognition from Trajectories and Correlated Regions

We show the model architecture in detail, and explain the architecture of different parts.

Model architecture

We summarize the architecture in Table 8, and show the details per module. The feature extractor is based on ResNet-18 (He et al. 2016) with four same ResNet-18 blocks. The trajectory module uses 1×1 convolution in each stage of the feature extract to encode the location map. The encoded location maps are further combined with the temporal attention results in each stage. The correlation module utilizes the dynamic attention mechanism in the main paper, and combines the results with the trajectory module as stated in the main paper. The BiLSTMLayer comes from the official implementation of Pytorch.

Modules	Parameter
Extractor Backbone	Conv3d(3, 64, kernel size=(1,7,7), stride=(1,2,2), padding=(0,3,3),bias=False) BatchNorm3d(64) ReLU(inplace=True) MaxPool3d(kernel size=(1,3,3), stride=(1,2,2), padding=(0,1,1)) ResNet-18 block 1, 64 ResNet-18 block 2, 128, stride=2 ResNet-18 block 3, 256, stride=2 ResNet-18 block 4, 512, stride=2 AvgPool2d(7, stride=1) Linear(512, 1000)
Trajectory module	Conv2d(in channels=20, out channels=32, kernel size=1, stride=1, padding=0) BatchNorm3d(32) ReLU(inplace=True) Conv2d(in channels=32, out channels=64, kernel size=1, stride=1, padding=0) BatchNorm3d(64) ReLU(inplace=True) Temporal Attention(dimension=64, number of heads = 8, dropout=0.0, bias=True) BatchNorm3d(64) Conv2d(in channels=20, out channels=64, kernel size=1, stride=1, padding=0) BatchNorm3d(64) ReLU(inplace=True) Conv2d(in channels=64, out channels=128, kernel size=1, stride=1, padding=0) BatchNorm3d(128) ReLU(inplace=True) Temporal Attention(dimension=128, number of heads = 8, dropout=0.0, bias=True) BatchNorm3d(128) Conv2d(in channels=20, out channels=128, kernel size=1, stride=1, padding=0) BatchNorm3d(128) ReLU(inplace=True) Conv2d(in channels=128, out channels=256, kernel size=1, stride=1, padding=0) BatchNorm3d(256) ReLU(inplace=True) Temporal Attention(dimension=256, number of heads = 8, dropout=0.0, bias=True) Conv2d(in channels=20, out channels=256, kernel size=1, stride=1, padding=0) BatchNorm3d(256) ReLU(inplace=True) Conv2d(in channels=256, out channels=512, kernel size=1, stride=1, padding=0) BatchNorm3d(512) ReLU(inplace=True) Temporal Attention(dimension=512, number of heads = 8, dropout=0.0, bias=True)
Correlation module	Dynamic attention(dimension=64, number of heads = 8, dropout=0.0, bias=True) Dynamic attention(dimension=128, number of heads = 8, dropout=0.0, bias=True) Dynamic attention(dimension=256, number of heads = 8, dropout=0.0, bias=True) Dynamic attention(dimension=512, number of heads = 8, dropout=0.0, bias=True)
BILSTM	BiLSTMLayer(rnn type='LSTM', input size=1024, hidden size=1024,num of layers=2, bidirectional=True)
Classifier	Linear(1024, 2001)

Table 8: **Network architecture** of our TCNet.