

Appendix for "Siformer: Feature-isolated Transformer for Efficient Skeleton-based Sign Language Recognition"

1 Qualitative examples of kinematic rectification

The qualitative examples provided in Figure 1 offer insights into the efficacy of our kinematic rectification approach in refining the skeletal representation of sign glosses. In column (a), we observe the initial values of keypoint coordinates before any rectification is applied. This raw data provides a baseline for comparison and highlights the inherent variability and noise present in skeletal representations. Column (b) visualises the skeletal representations after abduction-adduction rectification ($\alpha = 0.4$) is applied. This step aims to mitigate errors associated with lateral movement of the fingers, enhancing the alignment and precision of keypoint positions. Column (c) presents the skeletal representations after both abduction-adduction and flexion-extension rectifications ($\alpha = 0.4$) are applied. By reducing both lateral and angular deviations in movement, this rectification process yields refined keypoint coordinates that more accurately reflect the intended gestures and movements. While body pose keypoints exhibit relatively consistent and precise estimation, hand keypoints demonstrate greater variability and complexity due to the intricate nature of hand gestures in SLR contexts. Recognising this disparity, our kinematic rectification approach is designed to address the challenges associated with hand gestures.

2 Quantifying the performance of input-adaptive inference mechanism

We quantify the number of input samples passing through fewer layers than the defined layer number under the input-adaptive inference mechanism. The experiments employ a patience value of one, utilise the C1 configuration (detailed in the main paper for Siformer), and incorporate non-trained internal classifiers. Varying numbers of randomly selected samples from one of the two benchmark datasets are used for each test group. From Table 1, we can observe that the number of input samples going through fewer layers varies based on the number of inputs used in the experiments. As demonstrated in our previous experiments, the input-adaptive inference mechanism has proven effective in improving performance, regardless of the choice of patience value. We seek to gain deeper insight into the source of this improvement through quantification. The best-case scenario for performance improvement occurs when the internal classifiers make all decisions correctly. Taking the experiments conducted on 800 samples from WLASL800 as an example, the best-case accuracy improvement can be calculated as the number of samples undergoing fewer layers over the total number of samples, resulting in a 5.13% improvement when expressed as a percentage. According to our previous experiments, the actual improvement is 1.12% (refer to Table 2 and Table 3 in the main paper), suggesting that 21.83% of the detected samples can correctly exit earlier along the defined computational path. This discrepancy is reasonable considering the randomness of the testing samples for this experiment and the capacity of the

internal classifiers, which solely apply linear transformations to the input data. To mitigate the potential drawbacks associated with the non-trained internal classifiers, We conduct experiments on both the trained internal classifiers and brand new internal classifiers, revealing that the brand new internal classifiers provide slightly better performance (see Table 2).

Table 1: Quantifying the effectiveness of input-adaptive inference with a patience value of 1 and C1 configuration.

Dataset	Total samples count	Early exit cases count
WLASL100	800	41
	2400	131
	4000	195
LAS64	640	5
	1920	17
	3200	32

Table 2: Performance analysis with trained internal classifiers and non-trained internal classifiers on the WLASL100 dataset.

Encoders	Top-1 Accuracy (%)
(+) Non-trained new internal classifiers	86.60
(+) Trained internal classifiers	85.57

3 Detailed visualisation of Siformer

In the supplementary materials folder, we provide the saved Siformer model. Figure 2 presents an example showing a detailed partial view of the architecture. For a deep understanding of Siformer’s architecture, import the file named "Siformer.pth" into a visualisation tool accessible at ¹.

4 Effectiveness of positional encoding

In addition to the learnable frame-wise positional encoding (detailed in the main paper), we conduct experiments with two alternative positional encoding methods: absolute positional encoding with sinusoids [3] and learnable element-wise positional encoding. These methods offer distinct approaches to embed positional information, each with unique characteristics and implications for sequence modeling. The learnable element-wise positional encoding method utilises the same equation as the frame-wise encoding, as described by Equation 6 in the main paper. However, in the case of element-wise, each skeletal data within a frame is assigned a unique positional value ($P_{i,i} \neq P_{i,j}$), while skeletal data corresponding to the same keypoint across different frames share identical positional values ($P_{i,i} = P_{j,i}$). This approach ensures that positional values vary within a frame but remain consistent across frames.

¹<https://netron.app/>

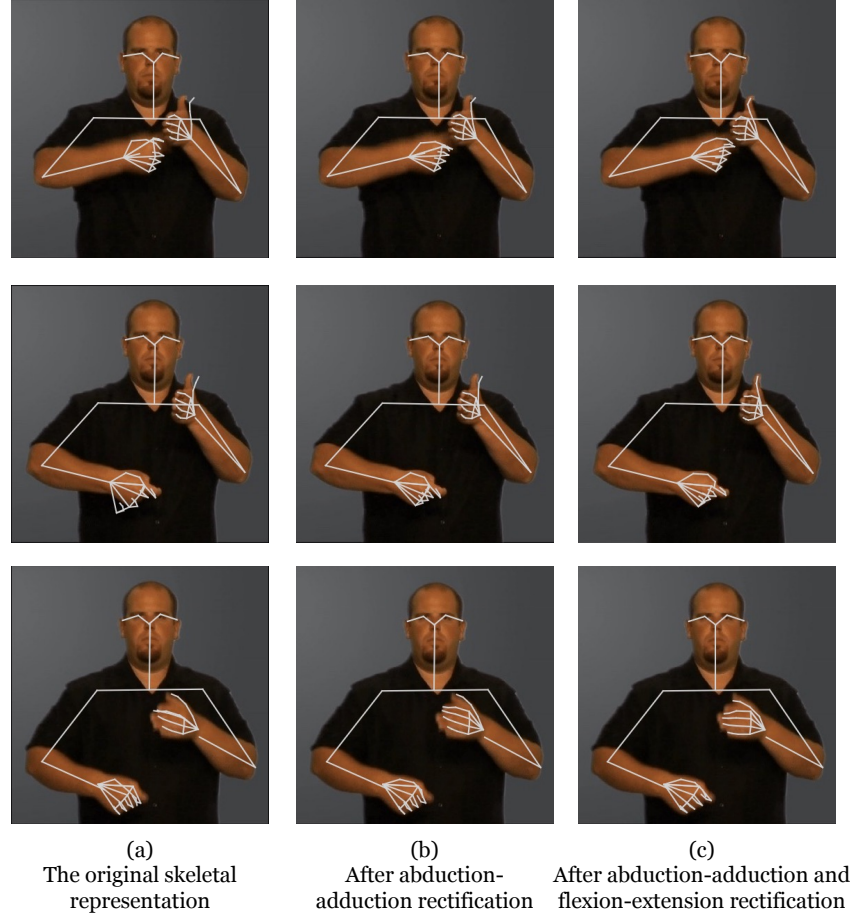


Figure 1: Qualitative examples of kinematic rectification

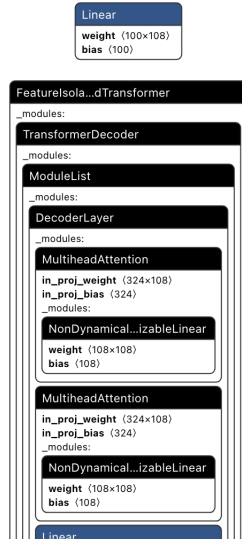


Figure 2: A partial view of Siformer in detail.

Both the input data X and the learnable positional embedding matrix P share the same dimension. Initially, P contains random values

within the range of $[0, 1)$, and it is jointly updated as one of the model parameters during the training process. On the other hand, the absolute positional encoding aims to embed absolute positional information without considering the sequential nature of the data. This encoding scheme does not differentiate between frames but assigns unique positional values to each skeletal data or element in the input sequence. Experimental results illustrated in Table 3 reveal that the learnable frame-wise positional encoding achieved the best performance among the three encoding methods. This finding suggests that embedding sequence information at the frame level plays a crucial role in enhancing the model performance compared to encoding positional information at the element level or using absolute positional encoding.

Table 3: Performance analysis with different positional encoding methods on the WLASL100 dataset.

Positional encoding	Top-1 Accuracy (%)
Learnable element-wised positional encoding	85.50
Learnable frame-wised positional encoding	86.50
Absolute positional encoding	80.00

5 Effectiveness of pyramid distilling

As a natural consequence of the ProbSparse self-attention mechanism detailed in the main paper, the encoders' feature map may contain redundant combinations of values V . To manage this issue and prioritise superior combinations with dominant features, the author [4] of the ProbSparse self-attention mechanism employ a distilling operation. This operation aims to create a focused attention feature map in the subsequent layer by selectively emphasising key information. The distilling procedure, as described in [4], involves forwarding from the j -th layer to the $(j + 1)$ -th layer:

$$X_{j+1}^t = \text{MaxPool}(\text{ELU}(\text{Conv1d}([X_j^t]_{AB}))) \quad (1)$$

The attention block $[\cdot]_{AB}$ comprises ProbSparse self-attention mechanism along with essential operations designed to enhance feature extraction and representation learning. Specifically, it incorporates $\text{Conv1d}(\cdot)$, which applies 1-D convolutional filters with a kernel width of 3, followed by the ELU activation function [2]. A max-pooling layer is introduced with a stride of 2 to downsample the input X_j^t into its half slice after stacking a layer. To enhance the robustness of the distilling operation, a strategy is implemented to construct replicas of the main stack with halved inputs. This involves progressively reducing the number of self-attention distilling layers, akin to a pyramid structure, while ensuring alignment of output dimensions. As a result, all stack outputs are concatenated to form the final hidden representation of the encoder.

However, the results presented in Table 4 reveal that the approach to distal features does not yield any improvement in performance. Instead, it leads to an increase in computational power requirements for the distilling operation and a degradation in overall SLR performance. This outcome suggests that while the concept of distilling information from distal features may hold theoretical promise, its practical implementation may introduce complexities and inefficiencies that outweigh any potential benefits. The increase in computational power required for the distilling operation indicates a significant overhead that may not be justified by the marginal gains in performance, if any.

Table 4: Ablation study of the pyramid distilling on the WLASL100 dataset.

Encoder	Top-1 Accuracy (%)
(-) Pyramid distilling	86.50
(+) Pyramid distilling	84.43

6 Data augmentation and normalisation

We include data augmentation and normalisation methods from [1]. These methods are directly applied to the extracted keypoint coordinates. Among the four data augmentation methods (visually illustrated at 3), one is randomly selected and applied to the training data; each training data has a 50% chance of being subjected to the selected data augmentation method. Gaussian noise is applied differently; once it is applied, it affects each training data. The normalisation process involves separating the space between the hands and body for each training data based on the boundary boxes of the hands and body. The part-based normalisation process is performed before data augmentation, and it is more effective in terms

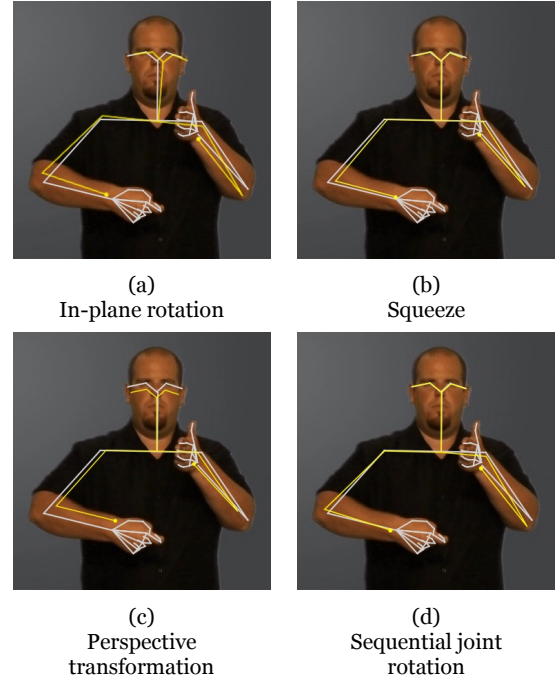


Figure 3: Qualitative examples of individual augmentations denoted by yellow skeletal representations.

Table 5: Ablation study of data augmentations and normalisation with Siformer under the WLASL100 dataset.

Normalisation	Augmentations	Top-1 Accuracy (%)
×	×	81.75
✓	×	86.25
✓	Rotate	86.00
✓	Squeeze	86.25
✓	Perspective transformation	85.75
✓	Arm joint rotate	85.88
✓	All	86.25
✓	All + Gaussian noise	86.50

of performance improvement (see Table 5) compared to the data augmentation methods. While these data augmentation methods yield slight performance improvement based on the WLASL100 dataset, they equip our model with improved capacity to handle variations and enhance robustness under non-ideal conditions.

References

- [1] Matyáš Boháček and Marek Hruš. 2022. Sign pose-based transformer for word-level sign language recognition. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 182–191.
- [2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR* (2016).
- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [4] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.