ZIPSIGN: TRAINING-FREE SPATIO-TEMPORAL TOKEN COMPRESSION FOR SIGN LANGUAGE TRANSLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Sign language translation (SLT) models have made steady progress by improving visual representation learning and translation modeling. However, existing SLT models usually process redundant visual tokens across spatial and temporal dimensions, leading to substantial computational costs and limiting their deployment in resource-constrained and real-time scenarios. In this work, we propose ZipSign, to the best of our knowledge, the first training-free spatio-temporal token compression method for sign language translation. ZipSign adopts a two-stage token compression strategy. The first stage performs similarity-based spatial token merging, which preserves sign-critical hand and face regions while merging redundant background tokens within each frame. The second stage performs confidence-guided temporal token pruning, which removes less informative frame-level visual tokens before they are fed into the translation module. ZipSign introduces no additional learnable parameters and can be directly plugged into existing SLT models during inference. Extensive experiments on standard SLT benchmarks demonstrate that ZipSign effectively reduces computational cost and inference latency while maintaining translation performance.

1 INTRODUCTION

Sign language translation (SLT) aims to translate continuous sign language videos into natural language sentences and is a promising technology to bridge the communication gap between deaf and hearing people Organization et al. (2018). Compared with general video understanding tasks, sign language unfolds across space and time through coordinated hand gestures, facial expressions, and body movements, where different visual cues interact continuously to convey complete linguistic meaning, making SLT particularly sensitive to fine-grained spatio-temporal visual cues Stokoe (1980); Meier (2002).

Recent advances have significantly improved SLT performance by incorporating stronger visual encoders Chen et al. (2022b); Zhou et al. (2023); Jiao et al. (2024); Li et al. (2025), temporal modeling modules Camgoz et al. (2020); Yin & Read (2020); Li et al. (2020); Gan et al. (2021), and large language models Gong et al. (2024); Wong et al. (2024); Liang et al. (2024); Chen et al. (2024b); Jang et al. (2025). Despite this progress, the growing modeling capacity of SLT systems is accompanied by practical efficiency challenges. Spatially, many regions within each frame correspond to background or weakly informative visual content, whereas sign-critical regions are usually concentrated around the hands and face. As illustrated in Figure 1, treating all spatial tokens equally may lead to unnecessary computation on redundant background regions. Temporally, sign videos contain repeated or slowly changing visual patterns, such as hold phases and transition intervals with limited semantic variation. As shown in Figure 2b and 2c, adjacent frame-level representations exhibit high similarity. Traditional SLT either retains all frame-level tokens or reduces them through fixed position sampling or fixed window pooling, as shown in Figure 2a. Retaining all tokens incurs unnecessary computation, whereas these content-agnostic reduction strategies may discard key sign language frames, thereby limiting their deployment in real-time or resource-constrained scenarios Yin et al. (2021); Gan et al. (2023). However, most existing SLT research primarily focuses on improving translation accuracy, while inference efficiency remains less systematically explored. Existing studies have started to consider token-efficient Yu et al. (2023); Low et al. (2025) or compact SLT representations Ye et al. (2024); Chen & Tsai (2026), but they usually require redesigning or retraining the model with additional training objectives, which limits their applicability to already

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

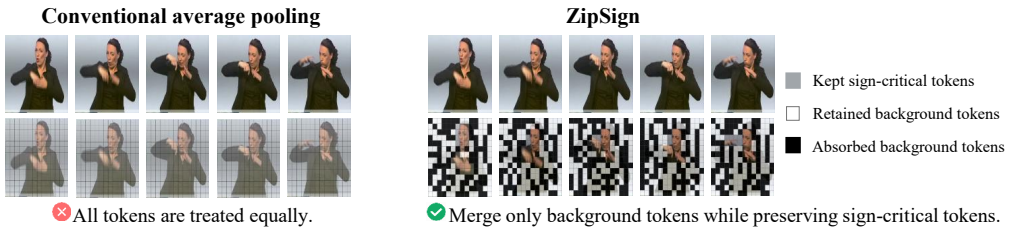
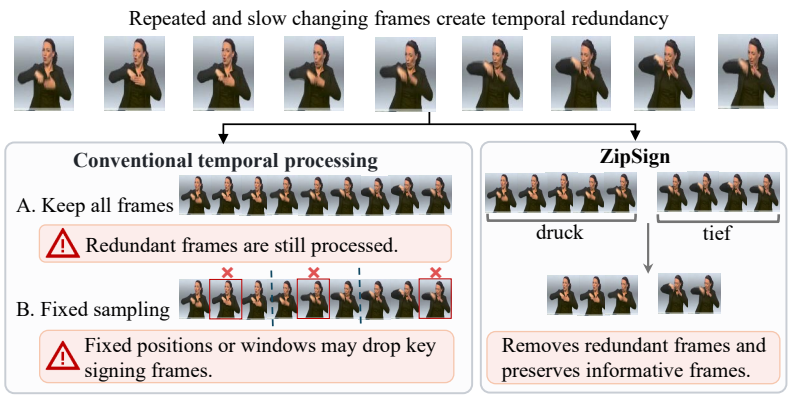
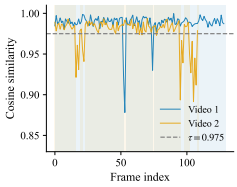


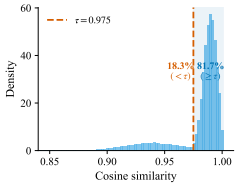
Figure 1: Spatial redundancy within frames.



(a) Comparison between conventional temporal processing and ZipSign confidence-guided temporal pruning.



(b) Adjacent frame similarity.



(c) Distribution of adjacent frame similarity.

Figure 2: Temporal redundancy across frames.

trained SLT models. This motivates an important challenge: **Can we reduce redundant visual tokens in existing SLT models without retraining them, while preserving the fine-grained visual cues required for accurate translation?**

Training-free token reduction has been widely explored in vision and multi-modal models, offering useful inspiration for efficient SLT. These methods typically exploit visual token redundancy through similarity-based merging Bolya et al. (2022); Choi et al. (2024); Hyun et al. (2025), saliency- or attention-based pruning Ye et al. (2025); Zhang et al. (2024a), or hybrid merging and pruning strategies Zhong et al. (2025); Tao et al. (2025); Huang et al. (2025), demonstrating that inference can be accelerated without retraining the original model. In practice, directly applying these general token merging or pruning strategies to SLT is not the best option. First, visually similar features may represent different linguistic information. For example, two short signing segments may exhibit nearly identical body postures and hand locations while differing only in handshape. Such subtle differences distinguish different lexical, but can be obscured by similarity-based merging. Second, SLT is sensitive to small but linguistically important visual signals. Hand regions and facial expressions often occupy only a small spatial area, yet subtle variations in these regions may change sentence meaning. Generic saliency- or attention-based methods are not explicitly designed to preserve these sign-specific regions and therefore provide no guarantee that all such cues will survive, especially under a limited token budget. Third, general video understanding tasks such as VideoQA Lei et al. (2020); Kim et al. (2020) usually only require a few key segments related to the question. In contrast, SLT is a semantically dense continuous temporal task, general temporal compression method may destroy the complete temporal order and contextual continuity required for translation. Therefore, efficient SLT requires token compression that is not only training-free, but also aware of sign-specific visual structures.

To address this challenge, we propose ZipSign, to the best of our knowledge, the first training-free spatio-temporal token compression framework that can be directly plugged into existing SLT models for fast inference. ZipSign adopts a two-stage compression strategy to reduce visual token redundancy from both spatial and temporal dimensions. Spatially, it preserves sign-critical hand and face

tokens and performs norm-directed similarity merging only among redundant background tokens within each frame. Temporally, it partitions the temporal sequence into sign segments according to gloss or pseudo-gloss predictions, retains the highest-confidence visual token from each segment together with its neighboring tokens to preserve local motion context, and fills the remaining token budget with the most confident unselected tokens. In this way, ZipSign improves inference efficiency while preserving translation performance as much as possible.

Our contributions are summarized as follows:

- To the best of our knowledge, we present the first training-free spatio-temporal token compression method for sign language translation.
- Our approach adopts a two-stage design that merges spatial tokens before frame-level visual tokens are formed and prunes temporal visual tokens before they enter the translation module. This design is simple and applicable to gloss-based and gloss-free SLT models.
- We evaluate ZipSign on representative SLT models and show that it improves inference efficiency while maintaining translation quality, demonstrating its practicality as a plug-and-play acceleration module for SLT.

2 Related Work

2.1 Sign Language Translation

Recent SLT methods have made steady progress by improving visual representation learning Chen et al. (2022b); Wong et al. (2024), temporal modeling Camgoz et al. (2020); Chen et al. (2022b), and cross-modal alignment Zhao et al. (2024); Zhou et al. (2023). Existing approaches generally follow either gloss-based or gloss-free paradigms. Gloss-based SLT methods use gloss annotations as intermediate linguistic supervision to align visual features with gloss sequences Camgoz et al. (2020); Chen et al. (2022a). Representative models such as TwoStream Chen et al. (2022b) exploit both RGB videos and keypoint sequences to capture complementary visual cues. In contrast, gloss-free methods directly learn the mapping from sign videos to target sentences Chen et al. (2022a); Shi et al. (2022); Lin et al. (2023). Some recent approaches further introduce visual-language pretraining Zhou et al. (2023); Jiao et al. (2024); Jiang et al. (2024), pseudo-gloss supervision Wong et al. (2024); Guo et al. (2026), or large language models Gong et al. (2024); Chen et al. (2024b); Kim et al. (2025); Hwang et al. (2025) to improve translation quality. For example, Sign2GPT Wong et al. (2024) leverages pretrained vision language models with lightweight adapters, and uses automatically extracted pseudo-glosses to enhance sign representation learning without requiring manual gloss annotations.

Despite these advances, efficiency remains underexplored in SLT. Sign videos usually contain redundant visual information both within each frame and across the temporal sequence, which makes inference costly, especially when long visual sequences are processed by the translation module. Most existing SLT studies mainly focus on translation accuracy, while training-free visual token reduction for SLT is still rarely studied. Recent token-efficient SLT attempts, such as SAGE Low et al. (2025), show that reducing visual sequence length can improve scalability, but they require modifying the model architecture and retraining with additional supervision. In contrast, our work targets a plug-in visual token compression setting, aiming to reduce redundant visual tokens in a training-free manner.

2.2 Training-Free Token Reduction in Vision and Multi-Modal Models

Token reduction has been widely studied in vision models to reduce the computational cost caused by excessive visual tokens Rao et al. (2021); Liang et al. (2022); Chen et al. (2024a); Zhang et al. (2024b). In image-based Vision Transformers, Token Merging (ToMe) Bolya et al. (2022) shows that redundant visual tokens can be merged at inference time according to embedding similarity, without requiring model retraining. For video models, token redundancy becomes more severe because visual tokens are produced not only within each frame but also across a long temporal sequence Huang et al. (2025). Recent studies therefore extend token reduction to video transformers Choi et al. (2024); Tao et al. (2025); Hyun et al. (2025). For example, vid-TLDR Choi et al. (2024) identifies salient regions and merges redundant background tokens. These methods show that

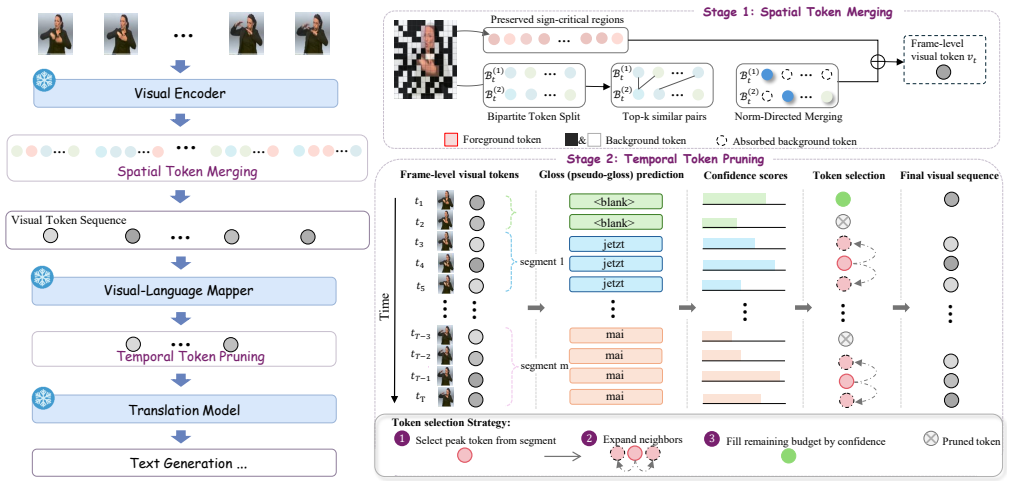


Figure 3: Overview of the proposed ZipSign framework.

reducing visual redundancy is especially important for video understanding, where long sequences bring substantial computational overhead. More recently, token reduction has also been explored in multi-modal and video LLMs Shen et al. (2026); Ye et al. (2025); Zhong et al. (2025); Huang et al. (2026; 2025). These methods combine similarity-based merging and importance-based pruning Zhong et al. (2025), preserve positional information during compression Huang et al. (2026), or select visual tokens according to their relevance to language queries Huang et al. (2025). Together, these studies demonstrate the potential of training-free token reduction for improving the efficiency of large vision language models.

However, existing token reduction methods are mainly designed for general image, video, or multi-modal understanding tasks. They usually do not consider the fine-grained linguistic cues in sign language, such as finger shape and facial expression. Directly applying generic token merging or pruning to SLT may weaken sign-critical regions or important frames, leading to degraded translation quality. Different from these methods, ZipSign preserves sign-critical regions during spatial background token merging within each frame and performs recognition confidence temporal token pruning to remove redundant frame-level visual tokens from the temporal sequence.

3 METHOD

ZipSign adopts a two-stage spatio-temporal token compression strategy. The spatial stage merges redundant background tokens within each frame while preserving hand and face critical regions, and the temporal stage prunes less informative frame-level tokens based on gloss or pseudo-gloss confidence. This approach aims to reduce spatio-temporal token redundancy while preserving translation performance as much as possible.

3.1 SPATIAL TOKEN MERGING

In SLT models, each video frame usually contains substantial spatial redundancy, including many background regions that are weakly related to translation semantics. However, SLT is highly sensitive to fine-grained visual cues such as hand movements and facial expressions, and directly compressing all spatial tokens may damage key semantic information. Therefore, we treat visual tokens around the hands and face as foreground information and keep them unchanged, only merging highly similar redundant tokens in the remaining background regions. The purpose of spatial token merging is not to reduce the number of frame-level tokens input to the translation module. After spatial merging, each frame is still represented by one visual token. This stage aims to reduce spatial redundancy, making the final feature representation more focused on sign-critical regions.

Before the frame-level representation is formed by spatial pooling, the visual backbone produces an intermediate feature map for the t-th frame. We flatten this feature map into spatial feature

cells $Z_t = [z_{t,1}, ..., z_{t,N}] \in \mathbb{R}^{N \times d}$, where N is the number of spatial cells and d is the feature dimension. In the following, we refer to these flattened spatial feature cells as spatial tokens. Spatial token merging is applied to these intermediate spatial tokens. We use keypoints to identify the hand and face regions, and denote the corresponding foreground token index set as $\mathcal{F}_t$. The remaining tokens are treated as background tokens, denoted as $\mathcal{B}_t$. All subsequent token matching and merging operations are restricted to $\mathcal{B}_t$, leaving $\mathcal{F}_t$ unchanged. Following the training-free token reduction method ToMe Bolya et al. (2022), we splits $\mathcal{B}_t$ into two sets $\mathcal{B}_t^{(1)}$, $\mathcal{B}_t^{(2)}$ and performs the bipartite soft matching between the two sets to form token groups. the cosine similarity between background tokens i and j in the t-th frame is defined as:

$$s_t(i,j) = \frac{\mathbf{z}_{t,i}^\top \mathbf{z}_{t,j}}{\|\mathbf{z}_{t,i}\|_2 \|\mathbf{z}_{t,j}\|_2}, \quad i,j \in \mathcal{B}_t. \tag{1}$$

For each token $i \in \mathcal{B}_t^{(1)}$, we select its most similar token from $\mathcal{B}_t^{(2)}$:

$$j(i) = \arg\max_{j \in \mathcal{B}_t^{(2)}} s_t(i,j). \tag{2}$$

The top-k matched pairs with the highest similarity scores are then selected for merging, where k is determined by the spatial merging ratio. Unlike ToMe, which consistently merges from the source set $\mathcal{B}_t^{(1)}$ to the destination set $\mathcal{B}_t^{(2)}$, we adaptively determine the merging direction based on feature norms. Here, the feature norm is used as a proxy for token saliency. For each selected pair (i, j), we compare the ℓ_2 norms of their feature vectors, and the larger norm token as the destination token. When multiple tokens are absorbed into the same destination token, we first average the features of these absorbed tokens before fusing them with the destination token. Background tokens not selected for merging remain unchanged.

After merging, we convert spatial tokens into frame-level visual representations to maintain the same output interface as existing SLT backbones. Instead of average pooling, we perform weighted spatial average pooling over foreground tokens and kept background tokens:

$$\mathbf{v}_t = \frac{\alpha \sum_{i \in \mathcal{F}_t} \mathbf{z}_{t,i} + \sum_{i \in \mathcal{K}_t} \hat{\mathbf{z}}_{t,i}}{\alpha |\mathcal{F}_t| + |\mathcal{K}_t|}, \quad \alpha > 1. \tag{3}$$

Here, $\mathcal{K}_t$ denotes the kept background token set after merging, and $\hat{\mathbf{z}}_{t,i}$ denotes the corresponding kept background token. The weighting factor α, allowing the frame-level representation is biased toward sign-critical regions while preserving weak background context.

3.2 Temporal Token Pruning

After spatial merging, the input to the translation module consists of a sequence of frame-level visual tokens $\{\mathbf{v}_t\}_{t=1}^T$ for each video. However, many frames in sign videos are temporal redundant, especially during hold or transition phases. Therefore, pruning these redundant tokens before they enter the translation module can significantly reduce computational demands. Specifically, we use gloss or pseudo-gloss confidence to estimate the importance of each frame-level visual token.

Let $\mathbf{V} = \{\mathbf{v}_t\}_{t=1}^T$ denote the temporal visual tokens after spatial merging. For gloss-based SLT models, we use the gloss posterior probabilities produced by the recognition branch to estimate token importance. Specifically, the confidence score of the t-th token is defined as $c_t = \max_{g \in \mathcal{G}} p_t(g)$, which measures the confidence of the most likely non-blank gloss prediction at time step t. Here, $\mathcal{G}$ denotes the non-blank gloss vocabulary and $p_t(g)$ is the predicted probability of gloss g at frame t. The CTC blank label is excluded since it does not correspond to a specific gloss. For gloss-free models with an existing pseudo-gloss prediction head, we use the same idea. Then, consecutive tokens with the same non-blank prediction are grouped into segments $\mathcal{M}_m$, and the token with the highest confidence in each segment is selected as the key token:

$$f_m = \arg\max_{t \in M_m} c_t \tag{4}$$

We denote the set of selected key token indices as $I_{key} = \{f_m\}_{m=1}^M$. For pseudo-gloss based models, we follow the same segmentation strategy: each token is assigned to its most confident

pseudo-gloss prediction, consecutive tokens with the same pseudo-gloss prediction are grouped into a segments, and the highest confidence token in each segment is selected as the key token.

Given a retained ratio γ, the target number of retained temporal visual tokens is $R = \max(\lceil \gamma T \rceil, |I_{key}|)$. To preserve motion context around each key token, we further include its neighboring tokens:

$$\mathcal{N}_\delta(f_m) = \{t \in \mathbb{Z} \mid |t - f_m| \leq \delta,\ 1 \leq t \leq T\}. \tag{5}$$

where δ controls the neighborhood size. The initial candidate set is defined as

$$I_{cand} = \bigcup_{f_m \in I_{key}} \mathcal{N}_\delta(f_m). \tag{6}$$

If $|I_{cand}| > R$, we retain all key tokens and select the remaining $R - |I_{key}|$ tokens with the highest confidence scores from the candidate set after excluding the key tokens. If $|I_{cand}| < R$, we fill the remaining budget with the highest confidence tokens from the unselected positions. The final selected token indices are sorted in temporal order and denoted as I, while the remaining tokens are pruned. The compressed visual sequence is then formed as

$$\mathbf{V}' = \{\mathbf{v}_t \mid t \in I\}. \tag{7}$$

This sequence is fed into the translation module.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics. We evaluate ZipSign on two widely used SLT benchmarks, RWTH-PHOENIX-Weather 2014T (PHOENIX14T) Camgoz et al. (2018) and CSL-Daily Zhou et al. (2021), covering different languages and signing scenarios. PHOENIX14T is a German Sign Language dataset collected from weather forecast broadcasts, with 7,096 training, 519 development, and 642 test samples. It contains 1,066 gloss labels and a target vocabulary of 2,877 words. CSL-Daily is a Chinese Sign Language dataset collected from daily-life scenarios, containing 18,401 training, 1,077 development, and 1,176 test samples, with 2,000 gloss labels and a target vocabulary of 2,343 words.

Following standard SLT evaluation protocols, we report translation quality using BLEU-1/2/3/4 Papineni et al. (2002) and ROUGE-L Lin (2004). BLEU measures n-gram precision between generated and reference translations at different granularities, while ROUGE-L evaluates sequence-level similarity based on the longest common subsequence. To evaluate inference efficiency, we report three metrics: latency per sample, which measures the average inference time for each evaluated sample; estimated GFLOPs per sample, which measures the computational cost under the same computational environment; and speedup, which is computed based on the total evaluation latency relative to the baseline.

Baselines and Comparison Methods. To show that ZipSign improves the inference efficiency of SLT models while maintaining translation quality, we adopt TwoStream Chen et al. (2022b) and Sign2GPT Wong et al. (2024) as the baselines. They are representative gloss-based and gloss-free SLT models, respectively, and achieve competitive performance on SLT. Since ZipSign is a training-free plug-in module, we apply spatial token merging before forming frame-level visual representations and temporal token pruning before feeding the visual tokens into the translation module, and evaluate it without any additional training. Except for the reduced number of tokens, all experiments are conducted under the same evaluation settings of baselines. We compare ZipSign with three training-free token compression baselines: Uniform, MaxPool, and STTM Hyun et al. (2025). The first two methods only compress the temporal visual sequence. Specifically, Uniform sampling uniformly subsamples the temporal tokens with a fixed stride, while MaxPool reduces the sequence length by pooling tokens within local temporal windows. Both methods are inserted at the same location as our temporal token pruning module. STTM is a spatio-temporal token merging method originally developed for video LLMs. We adapt its two stages inserted at the corresponding spatial and temporal compression locations of ZipSign. For a fair comparison, all methods are configured to match the retained token ratios of ZipSign as closely as possible and we try to maintain similar GFLOPs.

Table 1: Results of ZipSign and token compression methods on the gloss-based TwoStream backbone across PHOENIX14T and CSL-Daily. The retained ratio denotes the temporal token keep ratio, while the spatial token keep ratio is fixed at 50%.

Dataset	Retained Ratio	Method	Dev						Test					
			B1	B4	R-L	ms/sample ↓	GFLOPs ↓	Speedup ↑	B1	B4	R-L	ms/sample ↓	GFLOPs ↓	Speedup ↑
PHOENIX14T	100%	TwoStream	53.60	27.56	53.13	535.22	211.25	1.00×	54.54	28.87	53.41	503.99	197.80	1.00×
	50%	+ Uniform	49.28	25.39	50.90	290.87	210.89	1.84×	49.67	25.59	51.05	271.25	197.35	**1.86**×
		+ MaxPool	49.34	24.50	51.25	312.78	210.87	1.71×	49.84	25.57	51.31	305.62	197.35	1.65×
		+ STTM Hyun et al. (2025)	50.27	24.67	50.71	323.00	210.76	1.66×	51.03	25.99	51.45	302.03	**197.22**	1.67×
		+ ZipSign (Ours)	**52.40**	**26.30**	**51.74**	**288.05**	**210.54**	**1.86**×	**53.00**	**27.02**	**52.43**	**270.73**	197.40	**1.86**×
	30%	+ Uniform	40.38	17.80	43.67	290.32	209.56	1.84×	41.41	18.30	44.51	**264.99**	196.15	**1.90**×
		+ MaxPool	41.45	19.20	45.57	313.00	209.60	1.71×	42.96	20.34	46.33	299.08	196.17	1.69×
		+ STTM Hyun et al. (2025)	42.02	20.31	46.07	308.39	**209.29**	1.74×	43.22	20.97	46.68	293.39	**195.89**	1.72×
		+ ZipSign (Ours)	**49.56**	**24.48**	**50.39**	**288.60**	209.94	**1.85**×	**50.21**	**25.27**	**51.07**	280.54	196.51	1.80×
CSL-Daily	100%	TwoStream	55.14	25.30	54.91	479.08	240.42	1.00×	55.13	24.91	55.18	496.97	251.00	1.00×
	50%	+ Uniform	48.68	20.63	49.68	**251.66**	**233.67**	**1.90**×	47.75	20.12	49.67	**281.80**	**239.62**	**1.76**×
		+ MaxPool	51.30	22.61	51.85	252.48	234.43	**1.90**×	51.08	22.62	51.95	284.41	241.84	1.75×
		+ STTM Hyun et al. (2025)	**54.53**	24.67	**54.32**	277.63	238.10	1.73×	**54.47**	24.13	**54.14**	286.46	248.47	1.74×
		+ ZipSign (Ours)	54.17	**24.71**	53.76	252.02	237.50	**1.90**×	54.25	**24.38**	54.08	285.53	240.17	1.74×
	30%	+ Uniform	28.30	9.44	35.04	**242.23**	**224.91**	**1.98**×	28.05	9.65	34.81	**272.89**	**232.05**	**1.82**×
		+ MaxPool	42.33	16.85	44.91	252.55	234.49	1.90×	42.42	16.70	44.92	284.61	242.02	1.75×
		+ STTM Hyun et al. (2025)	39.20	15.52	43.46	263.72	236.57	1.82×	39.06	14.85	42.98	274.70	246.91	1.81×
		+ ZipSign (Ours)	**51.09**	**22.62**	**52.16**	244.96	230.95	1.96×	**50.87**	**22.16**	**52.28**	273.73	240.02	**1.82**×

Table 2: Results of ZipSign and token compression methods on the gloss-free Sign2GPT backbone across PHOENIX14T and CSL-Daily. The retained ratio denotes the temporal token keep ratio, while the spatial token keep ratio is fixed at 50%.

Dataset	Retained Ratio	Method	Dev						Test					
			B1	B4	R-L	ms/sample ↓	GFLOPs ↓	Speedup ↑	B1	B4	R-L	ms/sample ↓	GFLOPs ↓	Speedup ↑
PHOENIX14T	100%	Sign2GPT	44.05	19.47	43.26	630.52	1224.68	1.00×	43.53	20.56	43.03	570.01	1130.74	1.00×
	50%	+ Uniform	43.20	19.11	42.50	**322.48**	1224.66	**1.96**×	42.71	19.77	41.95	**280.45**	1130.73	**2.03**×
		+ MaxPool	42.89	18.90	42.19	325.16	1224.67	1.94×	42.40	19.16	41.86	285.81	1130.73	1.99×
		+ STTM Hyun et al. (2025)	38.63	15.89	37.62	345.20	1223.51	1.83×	38.09	16.22	37.05	310.97	**1127.25**	1.83×
		+ ZipSign (Ours)	**44.23**	**19.61**	**43.41**	330.73	**1223.50**	1.91×	**42.94**	**20.04**	**42.25**	288.95	**1127.25**	1.97×
	30%	+ Uniform	39.73	16.81	38.95	**313.37**	1124.66	**2.01**×	40.32	17.42	38.80	**264.20**	1130.73	**2.16**×
		+ MaxPool	41.31	17.42	40.82	313.96	1124.66	**2.01**×	40.37	17.52	39.64	285.21	1130.72	2.00×
		+ STTM Hyun et al. (2025)	27.99	10.14	27.92	335.99	1223.50	1.88×	29.89	10.59	28.29	296.60	1127.25	1.92×
		+ ZipSign (Ours)	**42.54**	**19.57**	**41.20**	315.22	**1123.24**	2.00×	**42.43**	**19.30**	**41.20**	275.13	**1127.24**	2.07×
CSL-Daily	100%	Sign2GPT	37.69	12.59	37.94	388.57	837.33	1.00×	37.15	12.26	37.70	378.58	1064.64	1.00×
	50%	+ Uniform	36.15	11.52	36.62	**227.40**	837.34	**1.71**×	36.22	11.90	36.80	**220.26**	1064.63	**1.72**×
		+ MaxPool	35.84	11.23	35.85	232.88	837.33	1.67×	34.32	10.86	35.19	232.62	1064.63	1.63×
		+ STTM Hyun et al. (2025)	34.10	10.07	34.38	303.15	**837.32**	1.28×	33.49	10.15	34.35	287.78	**1062.31**	1.32×
		+ ZipSign (Ours)	**36.28**	**11.75**	**36.68**	227.99	**837.32**	1.70×	**36.61**	**12.19**	**36.94**	221.78	1064.62	1.71×
	30%	+ Uniform	31.68	8.72	32.04	223.51	837.32	**1.74**×	31.61	9.12	31.32	227.80	1064.62	1.66×
		+ MaxPool	30.55	8.90	31.80	234.02	837.32	1.66×	28.64	8.11	31.11	231.05	1064.62	1.64×
		+ STTM Hyun et al. (2025)	23.17	4.79	24.28	249.28	837.32	1.56×	22.24	4.68	23.77	250.80	**1062.30**	1.51×
		+ ZipSign (Ours)	**34.30**	**10.70**	**34.21**	**223.40**	**837.31**	**1.74**×	**32.29**	**9.54**	**33.04**	**224.00**	**1062.30**	**1.69**×

Implementation Details. Our method is applied during the inference stage of pre-trained SLT models, with all model parameters kept frozen. All experiments are implemented in PyTorch and conducted on NVIDIA RTX A6000 48GB GPUs. To obtain stable latency measurements, each evaluation run is assigned to a single GPU. Unless otherwise specified, the spatial token keep ratio is fixed at 50% in all experiments. For temporal token pruning, we evaluate all methods under retained ratios $\gamma \in \{0.30, 0.50\}$. *The detailed insertion locations of the two compression stages in each baseline are provided in the appendix.*

4.2 Main Results

Table 1 and 2 present the main results of different training-free token compression methods on TwoStream and Sign2GPT across PHOENIX14T and CSL-Daily. Overall, ZipSign achieves a more favorable balance between translation quality and inference efficiency than Uniform, MaxPool, and STTM across different backbones and datasets. On the gloss-based TwoStream backbone, at the 50% keep ratio, ZipSign reduces test latency by 46.3% and 42.5% on PHOENIX14T and CSL-Daily, respectively, while still preserving 93.6% and 97.9% of the original BLEU-4 performance. The advantage becomes more evident under the 30% keep ratio, where ZipSign improves BLEU-4 over the other compression methods by at least 4.30 and 5.46 on PHOENIX14T and CSL-Daily, respectively. ZipSign also generalizes well to the gloss-free Sign2GPT backbone. At the 50% keep ratio, ZipSign reduces test latency by 49.3% and 41.4% on PHOENIX14T and CSL-Daily, respectively, while

Table 3: Component ablation of ZipSign on the CSL-Daily test set. Merge and prune denote spatial merging and temporal pruning, respectively.

Method	Ratio	TwoStream				Sign2GPT			
		B4	R-L	ms/sample ↓	Speedup ↑	B4	R-L	ms/sample ↓	Speedup ↑
Baseline	100%	24.91	55.18	496.97	1.00×	12.26	37.70	378.58	1.00×
Only merge	50%	24.28	53.30	471.83	1.05×	12.07	**37.34**	354.41	1.07×
Only prune		23.01	52.68	**282.20**	**1.76**×	11.63	36.29	**214.62**	**1.76**×
ZipSign		**24.38**	**54.08**	285.53	1.74×	**12.19**	36.94	221.78	1.71×
Only merge	30%	**23.68**	**53.26**	457.21	1.09×	**10.61**	**34.04**	347.57	1.09×
Only prune		20.88	50.79	**267.46**	**1.86**×	9.23	32.09	**220.11**	**1.72**×
ZipSign		22.16	52.28	273.73	1.82×	9.54	33.04	224.00	1.69×

Table 4: Sign-critical region protection on TwoStream over PHOENIX14T.

Keep	Method	Dev		Test	
		B4	R-L	B4	R-L
50%	Random	22.56	47.06	22.59	47.31
	Uniform	25.74	51.96	26.70	52.42
	ToMe Bolya et al. (2022)	27.14	52.72	28.31	53.03
	Spatial merging(Ours)	**27.16**	**52.98**	**28.73**	**53.44**
30%	Random	10.36	33.14	11.86	34.47
	Uniform	25.29	50.41	25.74	50.39
	ToMe Bolya et al. (2022)	26.83	52.42	27.21	51.61
	Spatial merging(Ours)	**27.23**	**52.95**	**28.54**	**53.37**

Table 5: Latency of temporal token reduction on TwoStream over PHOENIX14T.

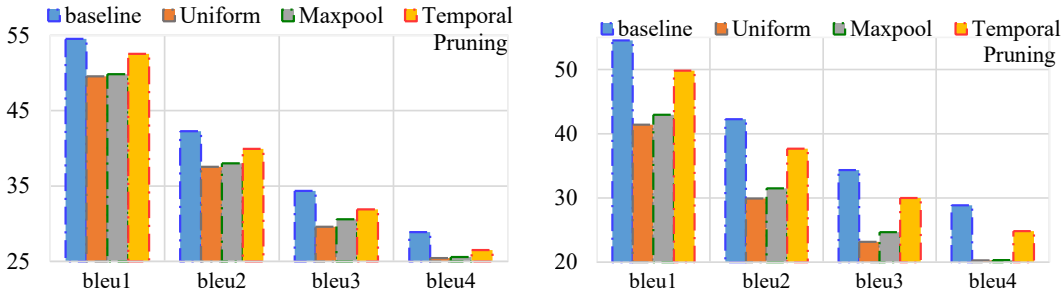
Retained Ratio	Method	ms/sample ↓
100%	Baseline	503.99
50%	Uniform	271.25
	MaxPool	305.62
	Prune (Ours)	**265.19**
30%	Uniform	264.99
	MaxPool	299.08
	Prune (Ours)	**263.69**

preserving 97.5% and 99.4% of the original BLEU-4 performance. In comparison, STTM causes more noticeable translation degradation, despite reducing the estimated GFLOPs. Under the 30% keep ratio, with comparable inference efficiency, ZipSign improves ROUGE-L over the other compression methods by at least 1.56 and 1.72 on the two datasets, respectively. These results suggest that generic spatio-temporal merging method does not fully preserve the fine-grained visual cues required by SLT. ZipSign can serve as a plug-in inference acceleration method for both gloss-based and gloss-free SLT models, reducing inference latency while better preserving translation quality.

To better interpret the efficiency results, we further analyze the relationship between estimated GFLOPs and measured latency. The reduction in average inference time is not always proportional to the reduction in estimated GFLOPs. In TwoStream,the visual backbone accounts for the majority of GFLOPs and remains largely unchanged after compression, whereas the actual latency is largely affected by the autoregressive generation of mbart. Our analysis shows that the visual backbone takes about 150-170 ms/sample, while the text generation accounts for a larger proportion of the end-to-end latency. Therefore, even with similar total GFLOPs, shortening the length of the visual sequence entering mBART can still reduce inference time.

4.3 Ablation Study

Effect of Two Components. We analyze the contribution of the two components in ZipSign on CSL-Daily, as shown in Table 3. Spatial Token Merging alone preserves translation quality well, since it mainly reduces redundant spatial tokens within each frame while protecting sign-critical regions. It achieves 23.68 BLEU-4 and 53.26 ROUGE-L at the 30% keep ratio, which is close to the baseline performance of 24.91 and 55.18 on the TwoStream baseline. However, its acceleration is less direct because the temporal sequence length remains unchanged. In contrast, Temporal Token Pruning directly shortens the temporal visual sequence, reducing the latency from 496.97 ms to 267.46 ms at the 30% keep ratio. Meanwhile, removing temporal visual tokens alone also causes a noticeable translation degradation, especially under the 30% keep ratio. These results indicate that spatial token merging helps preserve translation quality, while temporal token pruning provides the



(a) 50% retained tokens. (b) 30% retained tokens.

Figure 4: Effect of different temporal visual token compression strategies on TwoStream over PHOENIX14T test set.

main efficiency gain. Their combination enables ZipSign to accelerate inference while maintaining translation performance.

Effect of Sign-critical Region Protection. We compare our spatial token merging with random merging, uniform merging, and the ToMe strategy under the same spatial keep ratios. As shown in Table 4, at the 50% keep ratio, our method achieves 28.73 BLEU-4 and 53.44 ROUGE-L on the test set, slightly outperforming ToMe (28.31/53.03) and clearly surpassing uniform and random merging. The advantage becomes more pronounced at the 30% keep ratio: our method retains 28.54 BLEU-4 and 53.37 ROUGE-L, whereas ToMe and uniform merging decrease to 27.21/51.61 and 25.74/50.39, respectively, and random merging drops sharply to 11.86/34.47. These results indicate that treating all spatial tokens equally becomes increasingly harmful under aggressive compression. By preserving sign-critical regions and only merging background tokens, our method better maintains translation quality under different compression levels.

Effect of Temporal Token Selection. Figure 4 evaluates different temporal visual token compression strategies under the same retained ratio. For a fair comparison, all compared methods are applied before the visual tokens are fed into the translation module. Temporal token pruning consistently outperforms Uniform and MaxPool from BLEU-1 to BLEU-4. This suggests that uniform downsampling and local max pooling may discard semantically important signing cues, while based recognition confidence pruning better preserves important visual tokens.

We further compare the inference latency of different temporal token reduction strategies. As shown in Table 5, temporal token pruning achieves lower latency than Uniform and MaxPool. At the 50% retained ratio, it reduces the test latency to 265.19 ms/sample, compared with 271.25 ms/sample for Uniform and 305.62 ms/sample for MaxPool. At the 30% retained ratio, temporal token pruning also obtains the lowest test latency among the compressed methods. These results indicate that the performance gain of temporal token pruning does not come at the cost of inference efficiency. Instead, based recognition confidence token selection provides a better quality and efficiency balance for temporal visual token compression in SLT.

5 Conclusion

In this paper, we present the first training-free spatio-temporal token compression method for SLT, aiming to reduce redundant visual computation while preserving important signing cues. ZipSign reduces spatial redundancy by merging background tokens while preserving sign-critical regions, and reduces temporal redundancy by pruning less informative frame-level tokens based on gloss or pseudo-gloss confidence. It can be directly applied to frozen SLT models without additional training or architectural modifications. Experiments on representative gloss-based and gloss-free SLT models demonstrate that ZipSign improves inference efficiency while maintaining translation performance under different compression ratios. Compared with existing training-free token reduction strategies, ZipSign achieves a better efficiency-performance trade-off, demonstrating its effectiveness as a practical plug-in solution for efficient SLT deployment.

REFERENCES

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, Hermann Ney, and Richard Bowden. Neural sign language translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7784–7793, 2018.

Necati Cihan Camgoz, Oscar Koller, Simon Hadfield, and Richard Bowden. 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*, pp. 10023–10033, 2020.

Kuanwei Chen and Mengfeng Tsai. Towards compact sign language translation: Frame rate and model size trade-offs. *arXiv preprint arXiv:2605.09554*, 2026.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024a.

Yutong Chen, Fangyun Wei, Xiao Sun, Zhirong Wu, and Stephen Lin. A simple multi-modality transfer learning baseline for sign language translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5120–5130, 2022a.

Yutong Chen, Ronglai Zuo, Fangyun Wei, Yu Wu, Shujie Liu, and Brian Mak. Two-stream network for sign language recognition and translation. *Advances in Neural Information Processing Systems*, 35:17043–17056, 2022b.

Zhigang Chen, Benjia Zhou, Jun Li, Jun Wan, Zhen Lei, Ning Jiang, Quan Lu, and Guoqing Zhao. Factorized learning assisted with large language model for gloss-free sign language translation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 7071–7081, 2024b.

Joonmyung Choi, Sanghyeok Lee, Jaewon Chu, Minhyuk Choi, and Hyunwoo J Kim. vid-tldr: Training free token merging for light-weight video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18771–18781, 2024.

Shiwei Gan, Yafeng Yin, Zhiwei Jiang, Lei Xie, and Sanglu Lu. Skeleton-aware neural sign language translation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4353–4361, 2021.

Shiwei Gan, Yafeng Yin, Zhiwei Jiang, Lei Xie, and Sanglu Lu. Towards real-time sign language recognition and translation on edge devices. In *Proceedings of the 31st ACM international conference on multimedia*, pp. 4502–4512, 2023.

Jia Gong, Lin Geng Foo, Yixuan He, Hossein Rahmani, and Jun Liu. Llms are good sign language translators. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18362–18372, 2024.

Jianyuan Guo, Peike Li, and Trevor Cohn. Bridging sign and spoken languages: Pseudo gloss generation for sign language translation. *Advances in Neural Information Processing Systems*, 38:77471–77499, 2026.

Mouxiao Huang, Borui Jiang, Dehua Zheng, Hailin Hu, Kai Han, and Xinghao Chen. Ppe: Positional preservation embedding for token compression in multimodal large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Xiaohu Huang, Hao Zhou, and Kai Han. Prunevid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19959–19973, 2025.

Eui Jun Hwang, Sukmin Cho, Junmyeong Lee, and Jong C Park. An efficient gloss-free sign language translation using spatial configurations and motion dynamics with llms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3901–3920, 2025.

Jeongseok Hyun, Sukjun Hwang, Su Ho Han, Taeoh Kim, Inwoong Lee, Dongyoon Wee, Joon-Young Lee, Seon Joo Kim, and Minho Shim. Multi-granular spatio-temporal token merging for training-free acceleration of video llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23990–24000, 2025.

Youngjoon Jang, Haran Raajesh, Liliane Momeni, Gül Varol, and Andrew Zisserman. Lost in translation, found in context: Sign language translation with contextual cues. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 8742–8752, 2025.

Zifan Jiang, Gerard Sant, Amit Moryossef, Mathias Müller, Rico Sennrich, and Sarah Ebling. Signclip: Connecting text and sign language by contrastive learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9171–9193, 2024.

Peiqi Jiao, Yuecong Min, and Xilin Chen. Visual alignment pre-training for sign language translation. In *European Conference on Computer Vision*, pp. 349–367. Springer, 2024.

Hyounghun Kim, Zineng Tang, and Mohit Bansal. Dense-caption matching and frame-selection gating for temporal localization in videoqa. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4812–4822, 2020.

Jungeun Kim, Hyeongwoo Jeon, Jongseong Bae, and Ha Young Kim. Leveraging the power of mllms for gloss-free sign language translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21048–21058, 2025.

Jie Lei, Licheng Yu, Tamara Berg, and Mohit Bansal. Tvqa+: Spatio-temporal grounding for video question answering. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 8211–8225, 2020.

Dongxu Li, Chenchen Xu, Xin Yu, Kaihao Zhang, Benjamin Swift, Hanna Suominen, and Hongdong Li. Tspnet: Hierarchical feature learning via temporal semantic pyramid for sign language translation. *Advances in Neural Information Processing Systems*, 33:12034–12045, 2020.

Zecheng Li, Wengang Zhou, Weichao Zhao, Kepeng Wu, Hezhen Hu, and Houqiang Li. Uni-sign: Toward unified sign language understanding at scale. *arXiv preprint arXiv:2501.15187*, 2025.

Han Liang, Chengyu Huang, Yuecheng Xu, Cheng Tang, Weicai Ye, Juze Zhang, Xin Chen, Jingyi Yu, and Lan Xu. Llava-slt: Visual language tuning for sign language translation. *arXiv preprint arXiv:2412.16524*, 2024.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800*, 2022.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Kezhou Lin, Xiaohan Wang, Linchao Zhu, Ke Sun, Bang Zhang, and Yi Yang. Gloss-free end-to-end sign language translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12904–12916, 2023.

Jian He Low, Ozge Mercanoglu Sincan, and Richard Bowden. Sage: Segment-aware gloss-free encoding for token-efficient sign language translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4952–4961, 2025.

Richard P Meier. Why different, why the same? explaining effects and non-effects of modality upon linguistic structure in sign and speech. *Modality and structure in signed and spoken languages*, pp. 1–25, 2002.

World Health Organization et al. Addressing the rising prevalence of hearing loss. 2018.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Leqi Shen, Guoqiang Gong, Tao He, Yifeng Zhang, Pengzhang Liu, Sicheng Zhao, et al. Fastvid: Dynamic density pruning for fast video large language models. *Advances in Neural Information Processing Systems*, 38:123553–123581, 2026.

Bowen Shi, Diane Brentari, Gregory Shakhnarovich, and Karen Livescu. Open-domain sign language translation learned from online video. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6365–6379, 2022.

William C Stokoe. Sign language structure. *Annual review of anthropology*, pp. 365–390, 1980.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Dycoke: Dynamic compression of tokens for fast video large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18992–19001, 2025.

Ryan Wong, Necati Cihan Camgoz, and Richard Bowden. Sign2gpt: Leveraging large language models for gloss-free sign language translation. *arXiv preprint arXiv:2405.04164*, 2024.

Jinhui Ye, Xing Wang, Wenxiang Jiao, Junwei Liang, and Hui Xiong. Improving gloss-free sign language translation by reducing representation density. *Advances in Neural Information Processing Systems*, 37:107379–107402, 2024.

Weihao Ye, Qiong Wu, Wenhao Lin, and Yiyi Zhou. Fit and prune: Fast and training-free visual token pruning for multi-modal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22128–22136, 2025.

Aoxiong Yin, Zhou Zhao, Jinglin Liu, Weike Jin, Meng Zhang, Xingshan Zeng, and Xiaofei He. Simulslt: End-to-end simultaneous sign language translation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4118–4127, 2021.

Kayo Yin and Jesse Read. Better sign language translation with stmc-transformer. In *Proceedings of the 28th international conference on computational linguistics*, pp. 5975–5989, 2020.

Pei Yu, Liang Zhang, Biao Fu, and Yidong Chen. Efficient sign language translation with a curriculum-based non-autoregressive decoder. In *IJCAI*, pp. 5260–5268, 2023.

Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster. *arXiv e-prints*, pp. arXiv–2412, 2024a.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024b.

Rui Zhao, Liang Zhang, Biao Fu, Cong Hu, Jinsong Su, and Yidong Chen. Conditional variational autoencoder for sign language translation with cross-modal alignment. In *Proceedings of the aaai conference on artificial intelligence*, volume 38, pp. 19643–19651, 2024.

Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. Aim: Adaptive inference of multi-modal llms via token merging and pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20180–20192, 2025.

Benjia Zhou, Zhigang Chen, Albert Clapés, Jun Wan, Yanyan Liang, Sergio Escalera, Zhen Lei, and Du Zhang. Gloss-free sign language translation: Improving from visual-language pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20871–20881, 2023.

Hao Zhou, Wengang Zhou, Weizhen Qi, Junfu Pu, and Houqiang Li. Improving sign language translation with monolingual data by sign back-translation. In *CVPR*, pp. 1316–1325, 2021.

Appendix of "ZipSign: Training-Free Spatio-Temporal Token Compression for Sign Language Translation"

Paper ID 15490

6 AI USE STATEMENT

We used generative AI tools to assist with translation, drafting, and language editing of selected portions of the manuscript. We also sought feedback on the clarity and organization of the methodology section, including the terminology and explanations of model components. The authors reviewed and revised all AI-assisted text and checked the methodological descriptions against the actual implementation. We take full responsibility for the final content of this paper.

7 ADDITIONAL IMPLEMENTATION DETAILS

Inference-only evaluation. We implement ZipSign on two representative SLT backbones, TwoStream and Sign2GPT. All compression operations are enabled only during inference and use the same frozen checkpoints as their corresponding uncompressed baselines. No base-model parameter is fine-tuned, and no additional token reducer is trained. All experiments are implemented in PyTorch and conducted on NVIDIA RTX A6000 GPUs with 48 GB memory. Each evaluation job is assigned to a single GPU to obtain stable wall-clock measurements.

Backbone-specific insertion positions. ZipSign is applied at two visual-token interfaces. Spatial Token Merging is performed before frame-level visual tokens are formed. In TwoStream, it is inserted into the S3D/TwoStream visual backbone, where intermediate spatial feature tokens are merged before spatial pooling produces one representation for each frame. In Sign2GPT, it is inserted into the DINOv2 visual encoder, where redundant patch tokens are merged before the final frame-level visual representation is produced.

Temporal Token Pruning is performed at the translation-side visual-token interface. In TwoStream, it is applied after VLMapper and before the mBART translation module. In Sign2GPT, it is applied after the visual post-model and before the XGLM adaptor/cross-attention module. Uniform sampling and MaxPool are inserted at exactly the same temporal interface as our pruning module. Unless otherwise stated, temporal compression methods are evaluated at matched retained ratios $\gamma \in \{0.50, 0.30\}$. TwoStream uses recognition confidence from its CTC branch, whereas Sign2GPT uses confidence from its existing pseudo-gloss prediction head. Decoding follows the original protocol of each backbone: mBART decoding for TwoStream and autoregressive XGLM generation for Sign2GPT.

7.1 RESOLUTION-ADAPTIVE SPATIAL BUDGET ALLOCATION

In TwoStream, Spatial Token Merging is applied to intermediate S3D feature maps at three spatial resolutions, namely 28×28, 14×14, and 7×7. The fixed-budget variant uses the same background-token merge strength at every resolution:

$$r_l = r_{\text{base}}, \tag{8}$$

where r_l is the background merge ratio at feature level l. This strategy does not distinguish the larger spatial redundancy of shallow high-resolution features from the higher semantic density of deep low-resolution features.

The final TwoStream implementation instead reallocates the spatial compression budget according to feature-map resolution. Given a nominal global spatial keep ratio γ, the resolution-specific target

Table 6: Resolution-adaptive spatial merging budgets in the TwoStream backbone. The fixed-budget variant applies the same background merge ratio to all feature resolutions, whereas the adaptive variant redistributes the merging budget according to spatial resolution. Larger r_l indicates more aggressive background-token merging.

Keep ratio	Fixed r_{base}	Adaptive $r_{28\times28}$	Adaptive $r_{14\times14}$	Adaptive $r_{7\times7}$
50%	0.800	0.880	0.640	0.320
30%	1.000	1.000	0.857	0.571

keep ratio is

$$\widetilde{k}_l = \text{clip}(\gamma + \Delta_l, 0, 1)\,, \qquad \Delta_l = \begin{cases} -0.05, & 28\times28, \\ +0.10, & 14\times14, \\ +0.30, & 7\times7. \end{cases} \tag{9}$$

The corresponding background merge ratio is

$$r_l = \text{clip}\left(r_{\text{base}}\frac{1-\widetilde{k}_l}{1-\gamma}, 0, 1\right). \tag{10}$$

Here, $\text{clip}(x, 0, 1) = \min(\max(x, 0), 1)$. A larger r_l denotes more aggressive background-token merging. As shown in Table 6, the allocation moves most of the compression pressure to the 28×28 feature map, uses a moderate budget at 14×14, and protects the semantically denser 7×7 feature map. This strategy is resolution-adaptive rather than sample-adaptive: its budget is determined by the feature-map level and does not require a learned predictor or per-video optimization.

Shared spatial merging operation. The adaptive variant changes only the allocation of the merge budget. All other operations are identical to the fixed-budget spatial stage. Hand and face tokens identified from keypoints are preserved without merging, and bipartite cosine matching is performed only among background tokens. Each source token is paired with its most similar destination candidate. The feature norm determines the merging direction, with the larger-norm token retained as the destination. When several tokens are assigned to the same destination, their features are first averaged and then fused with the destination token.

After merging, the remaining spatial tokens are converted into one frame-level representation using the same weighted spatial pooling operation as defined in Eq. (3) of the main paper:

$$\mathbf{v}_t = \frac{\alpha\sum_{i\in\mathcal{F}_t}\mathbf{z}_{t,i} + \sum_{i\in\mathcal{K}_t}\widehat{\mathbf{z}}_{t,i}}{\alpha|\mathcal{F}_t| + |\mathcal{K}_t|}, \qquad \alpha > 1. \tag{11}$$

Here, $\mathcal{F}_t$ is the preserved hand-and-face token set and $\mathcal{K}_t$ is the retained background-token set after merging. Therefore, the pooling operation emphasizes sign-critical regions while retaining background context. The spatial stage remains training-free and outputs one $[B, T, C]$ frame-level token sequence, so it does not change the temporal sequence length passed to the subsequent modules.

7.2 Effect of Resolution-Adaptive Allocation

Table 7 isolates the effect of spatial budget allocation on PHOENIX14T using TwoStream. The two variants use the same insertion positions, protected regions, token matching, norm-directed merging, and weighted spatial pooling. At a nominal 50% spatial keep ratio, resolution-adaptive allocation improves test BLEU-4 from 27.01 to 28.73 and ROUGE-L from 52.75 to 53.44. The improvement is also retained under the more aggressive 30% setting. These results indicate that applying the same compression strength to every feature level can over-compress the semantically dense low-resolution representation, whereas shifting more of the merge budget toward high-resolution feature maps better preserves translation quality.

Table 7: Ablation of spatial budget allocation on TwoStream over PHOENIX14T. Both variants use otherwise identical spatial merging operations.

Keep	Budget allocation	Dev		Test	
		B4	R-L	B4	R-L
50%	Fixed across levels	26.07	52.26	27.01	52.75
	Resolution-adaptive	**27.16**	**52.98**	**28.73**	**53.44**
30%	Fixed across levels	26.19	52.56	26.89	52.48
	Resolution-adaptive	**27.23**	**52.95**	**28.54**	**53.37**

8 DISCUSSION

ZipSign addresses substantial spatial and temporal token redundancy in existing SLT models through a training-free compression framework. By preserving sign-critical hand and face regions during spatial merging and maintaining coverage of predicted sign segments during temporal pruning, it reduces inference cost while largely preserving translation quality across both gloss-based and gloss-free backbones. Despite its effectiveness, ZipSign has several limitations. The spatial stage relies on keypoints to identify sign-critical regions, so inaccurate or missing keypoints may reduce its benefit. The temporal stage depends on confidence signals provided by the underlying backbone, such as CTC or pseudo-gloss predictions, and may be affected when these signals are unreliable. In addition, the current spatial budget is resolution-adaptive but remains fixed across samples, without considering sample-specific spatial redundancy or signing complexity.