FastAdaSP: Multitask-Adapted Efficient Inference for Large Speech Language Model

Yichen Lu[1]* **Jiaqi Song**[1]* **Chao-Han Huck Yang**[2] **Shinji Watanabe**[1]
[1]Carnegie Mellon University [2]NVIDIA Research
{yichenl5, jiaqison, swatanab}@andrew.cmu.edu hucky@nvidia.com

Abstract

In this study, we aim to explore Multitask Speech Language Model (SpeechLM) efficient inference via token reduction. Unlike other modalities such as vision or text, speech has unique temporal dependencies, making previous efficient inference works on other modalities not directly applicable. Furthermore, methods for efficient SpeechLM inference on long sequence and sparse signals remain largely unexplored. Then we propose **FastAdaSP**, a weighted token merging framework specifically designed for various speech-related tasks to improve the trade-off between efficiency and performance. Experimental results on WavLLM and Qwen-Audio show that our method achieves the state-of-the-art (SOTA) efficiency-performance trade-off compared with other baseline methods. Specifically, FastAdaSP achieved **7x** memory efficiency and **1.83x** decoding throughput without any degradation on tasks like Emotion Recognition (ER) and Spoken Question Answering (SQA). The code will be available at https://github.com/yichen14/FastAdaSP

1 Introduction

Speech Language Models (SpeechLMs) have been an important role in the field of natural language processing and speech technology. Recent advancements (Hu et al., 2024; Chu et al., 2023; Sun et al., 2024b) have demonstrated significant capabilities in voice processing and audio understanding. Furthermore, GPT4-o (OpenAI, 2024) showcases conversational speech processing abilities, advancing the capability of LLMs toward various voice-interface applications. However, challenges related to inference latency and memory efficiency remain major bottlenecks, especially as multitask SpeechLMs grow larger, reaching up to 7 billion parameters. These challenges necessitate the development of more efficient inference methods.

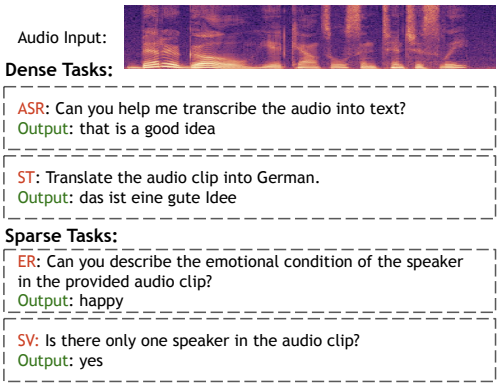

Figure 1: Examples of **Multitask SpeechLM** on dense (ASR, ST) and sparse (ER, SV) tasks

SpeechLMs are often capable of performing a wide range of speech or audio-related tasks. As shown in Figure 1, in our study, we categorize and define these tasks into two distinct classes: **Dense Tasks:** Nearly all input audio tokens are useful, such as in Automatic Speech Recognition (ASR) and Speech Translation (ST); **Sparse Tasks:** Tasks like Emotion Recognition (ER) and Speaker Verification (SV), where only a few tokens within the entire audio input contain the crucial information needed to perform the task.

The temporal dependencies in speech signals require efficient handling of long sequences, while the sparsity of relevant information demands precise extraction of crucial audio features. These unique properties make SpeechLM tasks distinct from other modalities like vision or text, especially when implementing token reduction techniques.

To address these issues and improve the efficiency of SpeechLM inference, we introduce **FastAdaSP**, a unified SpeechLM fast inference framework that incorporates multiple audio token reduction methods during the pre-filling stage tailored to different types of tasks. FastAdaSP does not require any additional training, making the entire framework more practical and easy to use. Our

*Equal Contributions.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 440–451
November 12-16, 2024

main contributions are as follows:

1. We introduce a new plug-and-play method for effectively selecting layers for audio token reduction operations on sparse tasks.

2. We study efficient inference methods specifically designed for both dense and sparse tasks on SpeechLMs and validate the effectiveness of our methods across multiple tasks.

3. To benchmark the task, previous token reduction methods, started from other modalities, have been investigated and analyzed in this emerging context of SpeechLM settings.

2 Related Work

Large Speech Language Models: SpeechLMs (Borsos et al., 2023; et al., 2023; Radhakrishnan et al., 2023; Sun et al., 2024b; Chu et al., 2023; Hu et al., 2024; Gong et al., 2024; Maiti et al., 2024; Lu et al., 2024) adopt a large pretrained language model (Touvron et al., 2023) as their base model and use audio encoder(s) (Radford et al., 2023; Chen et al., 2022; Hsu et al., 2021) to process raw audio input. Leveraging the language understanding and reasoning abilities of LLMs, SpeechLMs can perform various speech-related tasks. However, as SpeechLMs grow in size, inference latency and memory efficiency become problematic. Thus, research on cost-saving techniques is essential to address these challenges.

Efficient Inference in ASR: Recent studies (Zhu et al., 2024; Kim et al., 2022; Burchi and Vielzeuf, 2021) have focused on efficient inference for ASR models (Gulati et al., 2020; Kim et al., 2023) by progressively down-sampling the audio features in the audio encoder to reduce sequence length. However, these methods are specifically designed for the ASR task and do not generalize well to multitask settings for SpeechLMs.

Key-Value (KV) Cache Compression: In addition to the efficient inference methods for ASR, some of other works are focusing on compressing KV Cache to speed-up LLMs inference. Previous works such as StreamLLM (Xiao et al., 2024), H_2O (Zhang et al., 2023), LESS (Dong et al., 2024), LOOK-M (Wan et al., 2024) were designed to compress the text or vision KV cache during inference to overcome the limited KV cache size and accelerate the inference speed. However, KV cache compression techniques do not actually reduce the number of input tokens during the pre-filling stage. When a long video is input to a multimodal LLM, the extensive sequence of vision and audio tokens can exceed the context length limit of the backbone LLM, causing several issues. Moreover, this technique does not improve the latency of the pre-filling stage.

Token Reduction: To address these issues, extensive research has been conducted on token pruning techniques within Vision Language Models (VLMs). Recently, lots of token reduction works such as FastV (Chen et al., 2024), ToMe (Bolya et al., 2023), LLava-PruneMerge (Shang et al., 2024) focus on reducing the vision tokens to lower the computational costs through token eviction or merge. Besides the vision modality, A-ToMe (Li et al., 2023) applied the ToMe (Bolya et al., 2023) method to the audio modality in a Transformer-transducer model (Zhang et al., 2020) for ASR tasks only. However, token reduction methods for the audio modality in multitask SpeechLMs remain unexplored. Inspired by these previous works, our study primarily develops token reduction techniques that combine token merging and eviction for the audio modality in SpeechLMs during the inference process. We also explore the applicability of these methods to various speech-related tasks.

3 Methodology

In this section, we introduce the motivation and formulation of FastAdaSP, followed by our layer selection and task-specific design strategies for Multitask SpeechLMs. Note that, in our work, *audio tokens* refers to the audio features output by the multi-head attention block.

3.1 Preliminary

Speech Modality in Multitask SpeechLMs: During inference, VLMs often use only a *small portion* of visual information for reasoning and context understanding. However, SpeechLMs are capable of performing multiple tasks within a single model. For sequence-to-sequence dense tasks like ASR, it is crucial to consider "all audio tokens" to generate accurate transcriptions. In addition to dense tasks, SpeechLMs also need to perform sparse tasks such as ER and SQA, where only a few tokens in the input hold critical information for generating accurate predictions. Therefore, a more careful token reduction policy is necessary for SpeechLMs.

Pre-filling Phase of SpeechLMs: During the pre-filling phase of SpeechLMs, the raw audio sequence is usually processed by pre-trained audio

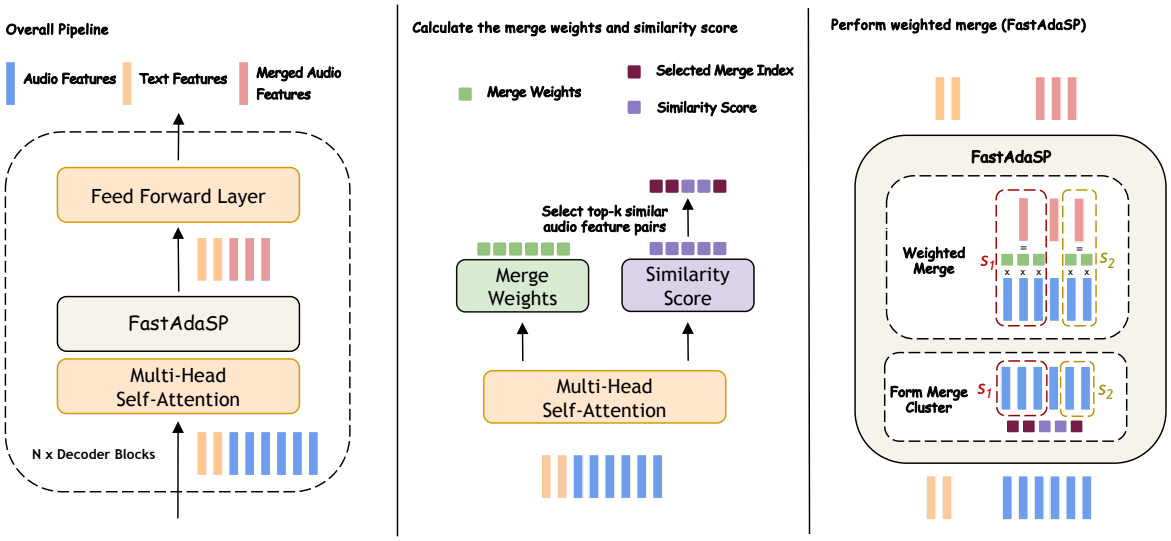

Figure 2: **FastAdaSP: Weighted Token Merge** of audio features in the decoder blocks of SpeechLMs

encoder(s) to extract the semantic and acoustic information into the embedding space $\mathbf{X}_{\text{audio}} \in \mathbb{R}^{L_{\text{audio}} \times D}$. Consider the text embedding of user instruction $\mathbf{X}_{\text{text}} \in \mathbb{R}^{L_{\text{text}} \times D}$, the input to the decoder blocks of SpeechLM is $\mathbf{X} \in \mathbb{R}^{L_{\text{prompt}} \times D}$, which represented as $\mathbf{X} = [\mathbf{X}_{\text{audio}}, \mathbf{X}_{\text{text}}]$. Here, L_{prompt} is the sum of audio embedding length L_{audio} and text embedding length L_{text}, and D is the model's hidden dimension.

In each self-attention block of the transfomer decoder layer, the query, key, value tensors can be derived by:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q,\ \mathbf{K} = \mathbf{X}\mathbf{W}_K,\ \mathbf{V} = \mathbf{X}\mathbf{W}_V\ , \quad (1)$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{D \times D}$ represents the matrix weights for query, key, and value layers, respectively. After this computation, the value of $\mathbf{K}, \mathbf{V}$ will be stored in the KV cache which will be used in the decoding phase. Then the self-attention output can be computed as:

$$\mathbf{X}_{\text{attention}} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}}\right)\mathbf{V}. \quad (2)$$

Decoding Phase of SpeechLMs: During the auto-regressive decoding phase of SpeechLMs, the KV cache is employed and updated for all the new generated tokens. At each step, the total key and value are calculated by using the previous stored $\mathbf{K}_{\text{cache}}$ and $\mathbf{V}_{\text{cache}}$ and the new input $\mathbf{X}_{\text{new}}$ as:

$$\mathbf{K} = [\mathbf{K}_{\text{cache}}, \mathbf{X}_{\text{new}}\mathbf{W}_K], \mathbf{V} = [\mathbf{V}_{\text{cache}}, \mathbf{X}_{\text{new}}\mathbf{W}_V]. \quad (3)$$

Equation 2 is used to calculate the attention output. During this stage, the KV cache grows linearly, and each new token significantly increases memory consumption and attention computation latency, especially when the generated sequence is very long.

3.2 FastAdaSP: Method

To accommodate both sparse and dense tasks in SpeechLMs, we designed a novel token reduction method with different strategies for each.

Weighted Token Merge: Dense tasks like ASR require most of the token information during inference, making direct token dropping from the attention output too aggressive and likely to result in the loss of critical information. Instead, merging similar audio tokens can eliminate redundant audio information while preserving essential content.

Token merge techniques in the vision modality require calculating the similarity between numerous pairs of image patches in the spatial domain to identify the most similar pairs for merging (Bolya et al., 2023). For audio signals, however, token merge in audio processing needs to operate in the temporal domain. This involves calculating the similarity along adjacent audio tokens pairs and merge a *cluster* of adjacent audio tokens for a sequence of audio features $A = (\mathbf{a}_i \in \mathbb{R}^D | i = 1, ..., L)$. For the audio features from 1 to $L-1$, we use the cosine similarity score between the adjacent audio token key state to determine their similarity:

$$p_i = \frac{\mathbf{K}_i^\top \cdot \mathbf{K}_{i+1}}{\|\mathbf{K}_i\|\|\mathbf{K}_{i+1}\|} \quad (4)$$

We then obtain an adjacent similarity score sequence $P = (p_i \in \mathbb{R} | i = 1, ..., L-1)$. After determining the number of tokens to merge, we select the top-k largest adjacent similarity indices to form the merge index list. Next, we loop through the merge index list, grouping multiple adjacent indices into a single merge cluster. Finally, we obtain m merging clusters $S = \{s_i | i = 1, ..., m\}$ where s_i represent a merging cluster which contains several adjacent audio tokens. Then we obtain the merge weights $W_{\text{merge}} = (\omega_i \in \mathbb{R} | i = 1, ..., L)$ for audio features $\mathbf{A}$ by:

$$\omega_i = (\sum^{H} \sum^{L_{\text{prompt}}} \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}}\right))_i \quad (5)$$

Where $L_{\text{prompt}} = L_{\text{audio}} + L_{\text{text}}$ represents the query length. H represents the number of attention head. Both audio and text features are utilized to calculate the overall cumulative attention score. By leveraging the interaction between text instructions and speech, we can determine the importance of audio tokens in the current context. The merged audio feature $\mathbf{a}_i^{\text{merge}}$ for each cluster s_i will be calculated as:

$$\mathbf{a}_i^{\text{merge}} = \frac{\sum^{|s_i|} \omega_j \mathbf{a}_j}{\sum^{|s_i|} \omega_j} \quad (6)$$

The overall procedure of the weighted token merge is shown in Figure 2. This method selects the relatively important tokens to keep and the redundant tokens to drop at that layer, effectively preserving as much information as possible while significantly reducing the number of tokens. For full details of the algorithm, please refer to A.3.

3.3 FastAdaSP: Strategies

Based above method, we designed two similar but slightly different strategies for dense and sparse tasks to achieve better performance:

Dense Task Strategy: For dense tasks, we designed an operation scheduler that smoothly merges tokens layer by layer to prevent aggressive token dropping in SpeechLM. We implemented a constant schedule to maintain a consistent merge ratio and a decay schedule that linearly decreases the merge ratio to zero at the final layer. Please refer to 4.3 for the ablation study of schedulers.

Sparse Task Strategy: For sparse tasks, a more aggressive token reduction method can be applied by merging tokens within a single layer. However, layer selection needs to be approached carefully as it significantly affects task performance. Therefore, we incorporate a *Transfer Entropy*(TE)-based layer selection method (Section 3.4) specially designed for sparse tasks.

3.4 Addtional Studies on Layer Selection

Recent token reduction works (Chen et al., 2024; Shang et al., 2024; Bolya et al., 2023; Li et al., 2023) often struggle with selecting appropriate layers for token reduction. Due to the difficulties in interpreting current auto-regressive transformer models, understanding the exact properties of different layers during inference is challenging. Consequently, previous works have relied on empirical studies to test various layers and reduction ratios. This approach is impractical and lacks generalization for actual deployment. Therefore, we aim to explore a justification to serve as a theoretical attempt of token reduction layer selection.

By definition, entropy can reflect the information carried out by each layer. Here, we take F as the feature output by the attention block which contains both audio and text features. Inspired by (Sun et al., 2022; Lin et al., 2024), we use the Gaussian distribution as the probability distribution to approximate the distribution of each channel in F. Thus, the entropy measurement of a single layer $H(F)$ can be defined as (for a more detail derivation, please refer to A.4):

$$H(F) \propto H_\sigma(F) = \sum_i \log[\sigma(F^i)] \quad (7)$$

Here, we calculate the entropy of each layer by summing the logarithm of the standard deviation(σ) of the each channels (audio tokens) in F. To assess the impact of weighted merge on a specific layer's contribution to the final output distribution, we calculate the Transfer Entropy to measure the information difference at the final layer based on the operation layer of our method. We define Transfer Entropy (TE_i.) for layer i. TE_i is equal to:

$$|H\left(\Phi\left(F_{\text{final}}; \mathbb{W}_{\text{final}}\right)\right) - H\left(F_{\text{final}} \mid \Phi\left(F_i; \mathbb{W}_i\right)\right)| \quad (8)$$

where $\Phi(\cdot;\cdot)$ represents the token reduction operation described in Section 3.2. It takes the layer feature F and merge weights $\mathbb{W}$ as input and outputs the features after weighted token merge. Then TE_i is the absolute difference between the final hidden states whether the token reduction operation is applied to layer i. The smaller the TE_i, the less the final information loss caused by the operation on layer i. We also analyze the effectiveness of our TE-based layer selection method in Sec. 4.2.

Model	Task	Dataset	Split	Metric
WavLLM (Hu et al., 2024)	Automatic Speech Recognition (ASR)	LibriSpeech (Panayotov et al., 2015)	dev | test	WER
	Speech Translation (ST)	Must-C (Di Gangi et al., 2019)	en-de	BLEU
	Emotion Recognition (ER)	IEMOCAP (Busso et al., 2008)	Session 5	ACC
	Spoken Language Answering (SQA)	MuTual (Cui et al., 2020)	test	ACC
Qwen-Audio (Chu et al., 2023)	Automatic Speech Recognition (ASR)	LibriSpeech (Panayotov et al., 2015)	dev | test	WER
	Speech Translation (ST)	CoVoST2 (Wang et al., 2020)	en-zh	BLEU
	Emotion Recognition (ER)	MELD (Poria et al., 2019)	test	ACC
	Audio Caption (AC)	Clotho (Drossos et al., 2020)	test	CIDEr | SPICE | SPIDEr

Table 1: **Task, dataset, and metrics** in the experiments

	ASR (WER% ↓)					ST (BLEU ↑)				
Full Token Baseline	2.25					21.56				
FLOPs Reduce	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
H_2O (Zhang et al., 2023)	2.25	2.46	3.37	5.60	10.20	20.42	20.12	19.99	18.64	17.36
Random Merge	2.32	2.51	3.08	8.10	77.42	20.73	19.34	18.23	15.69	9.24
Random Evict	2.53	4.34	9.23	34.80	172.52	20.34	19.03	17.36	14.10	8.59
A-ToMe (Li et al., 2023)	2.35	2.92	4.43	15.87	50.46	20.53	19.42	17.29	13.42	8.51
FastV (Chen et al., 2024)	2.37	4.94	13.84	51.10	185.79	21.17	20.18	18.98	16.36	10.07
FastAdaSP-Dense (Decay)	**2.27**	2.57	2.74	3.53	6.09	20.92	20.59	19.66	18.06	16.40
FastAdaSP-Dense (Constant)	**2.27**	**2.49**	**2.48**	**2.96**	**4.73**	**21.47**	**20.72**	**19.81**	**18.54**	**17.45**

Table 2: Comparison between FastAdaSP with other token reduction methods on WavLLM **dense tasks**

4 Experiments

4.1 Experiment Setting

Basic Settings: We use 1×V100 32GB GPU to conduct the task performance experiment. We also use 1×A100 80GB GPU and 1×H100 80GB GPU for long sequence system metric experiment. We choose WavLLM 7B (Hu et al., 2024) and Qwen-Audio 7B (Chu et al., 2023) for all the experiments. For each SpeechLM, we choose two *dense tasks* and two *sparse tasks* for experiments. Specifically, both models choose ASR and ST as dense task. For sparse task, we choose Emotion Recognition (ER) and Audio Caption (AC) on Qwen-Audio; ER and SQA on WavLLM. The full details of the dataset information and the evaluation metrics can be found in Table 1.

System Metrics: We use Theoretical FLOPs, Real Time Factor (RTF), Pre-filling and Decoding Latency (seconds per sentence), and Throughput (tokens per second) to measure the efficiency of our method under different token reduction rates. We calculate the RTF by:

$$\text{RTF} = \frac{T_{\text{Pre-filling}} + T_{\text{Decoding}}}{T_{\text{audio}}} \tag{9}$$

Where $T_{\text{Pre-filling}}$ and T_{Decoding} represents the pre-filling and decoding latency, T_{audio} represents the audio length (second per sentence).

4.2 Results and Discussion

In this section, we compare our method with other SOTA methods. Then, we demonstrate the impact of token reduction on system metrics. For the full experiments results, please refer to Appendix A.1.

Baselines: We selected several token reduction methods as our baselines. FastV (Chen et al., 2024) is a token eviction method based on attention scores for VLM. A-ToMe (Li et al., 2023) incorporates pair-wise merging techniques on the Transducer Model for ASR. We also test two other baselines method which randomly merge or evict tokens as the additional reference. Additionally, we applied our layer selection method to FastV and the two other random baselines since they do not have a clear layer selection strategy for speech tasks. Randomly choosing layers for these methods could result in completely failed decoding. Lastly, we evaluate the performance of the KV cache eviction method (H_2O) (Zhang et al., 2023) on SpeechLMs for reference. However, this method is primarily designed to accelerate multi-round generation, focusing on a different set of challenges and applications compared to our work.

Efficient Inference for Dense Tasks: We selected

	ER (ACC% ↑)					SQA (ACC% ↑)				
Full Token Baseline	72.80					67.60				
FLOPs Reduce	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
H_2O (Zhang et al., 2023)	72.32	72.60	73.73	72.11	72.36	67.10	68.4	68.00	67.55	65.40
Random Merge	72.76	72.44	72.19	72.28	72.52	67.40	68.00	67.40	67.95	68.30
Random Evict	73.08	71.71	72.44	72.03	72.28	67.65	67.35	68.35	67.80	67.50
A-ToMe (Li et al., 2023)	72.84	**72.68**	72.2	71.23	69.54	67.05	67.15	65.75	63.45	62.60
FastV (Chen et al., 2024)	72.76	72.52	71.55	71.47	70.66	67.45	67.25	**68.45**	68.10	67.95
FastAdaSP-Sparse	**73.16**	72.60	**73.73**	**73.65**	**73.65**	**67.65**	**68.05**	67.45	**68.45**	**68.70**

Table 3: Comparison between FastAdaSP with other token reduction methods on WavLLM **sparse tasks**

FLOPs Reduction %	Device	Real Time Factor ↓	Pre-filling Latency (s) ↓	Decoding Latency (s) ↓	Throughput (token/s) ↑
0.00	A100 80G	0.126	6.72	23.55	3.10
50.00		0.077	6.48	11.89	5.72
0.00	H100 80G	0.039	1.13	8.39	8.70
50.00		0.026	0.96	5.42	12.55

Table 4: **Long Sequence Computational cost experiments** on a 240s audio sample with a batch size = 5 on WavLLM using one A100 80GB GPU and one H100 80GB GPU. For the full results, please refer to Appendix A.1

Token Reduce %	Max Batch Size (not OOM)
Full Token Baseline	10
50	70

Table 5: **Memory Saving Experiments:** Approximate maximum batch size under 50% token reduction for WavLLM using a 240s audio sample on 1×A100 80GB.

FLOPs Reduce	TE	TE Rank	10%	20%	30%	40%	50%
Layer 2	2.20	4	54.78	54.30	54.06	52.91	52.10
Layer 9	2.17	3	**55.51**	54.30	53.61	53.30	51.50
Layer 12	2.29	5	54.75	53.96	53.44	52.72	48.35
Layer 15	2.11	2	53.98	54.06	53.02	50.57	-
Layer 3 (Selected)	2.06	1	55.17	**55.05**	**54.40**	**53.86**	**52.14**

Table 6: **Layer Selection Experiments:** Comparison on the performance between different layers on Qwen-Audio ER task (Full token baseline accuracy: 54.80%)

ASR and ST as the dense tasks in SpeechLM. As shown in Table 2, our method demonstrates a significantly better efficiency-performance trade-off compared to other token reduction methods. Notably, for the ASR task, we maintain only approximately 0.7% WER degradation up to a 40% FLOPs reduction ratio. Furthermore, we significantly improve upon the previous audio efficient inference baseline, A-ToMe, reducing the WER from 50.46% to 4.73% at a 50% FLOPs reduction rate. For the ST task, our method also maintain the best efficiency-performance trade-off with only approximate 4 BLEU score degradation on 50% FLOPs Reduce Rate.

Efficient Inference for Sparse Tasks: For the *Sparse Task* result in Table 3, our method not only surpasses most of the token reduction methods but also improves the original full token baseline from 67.6% to 68.7% accuracy for SQA and from 72.8% to 73.65% accuracy for ER. These experimental results demonstrate that sparse tasks can be enhanced by the token reduction method, which helps the model ignore redundant audio tokens in a more effective manner.

Computational Cost Analysis: We analyze our token reduction method across various system metrics and demonstrate efficiency improvements at a 50% token reduction rate. The results in Table 4 show that we achieved a **1.84x** increase in decoding throughput (from 3.10 tokens/s to 5.72 tokens/s) under A100 GPU and a **1.44x** throughput under H100 GPU. Further, our method can also decrease both pre-filling and decoding latency at about 4% and 50%, respectively.

Memory Saving Analysis: For memory efficiency in batch decoding settings, as shown in Table 5, our system can achieve approximately a **7x** increase in batch size after a 50% token reduction in practical deployment. These improvements demonstrate the significant potential of our token reduction method in enhancing both computational and memory efficiency for large-scale applications.

	ASR (WER% ↓)					ER (ACC% ↑)				
Full Token Baseline	2.25					72.80				
FLOPs Reduce	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
Average Merge	2.25	2.53	3.92	12.78	93.14	72.52	72.28	73	71.95	72.84
Weighted Merge	**2.25**	**2.44**	**3.25**	**10.51**	**90.24**	**73.16**	**72.6**	**73.73**	**73.65**	**73.65**

Table 7: **Average Merge vs. Weighted Merge.** The effectiveness of weighted merge method on WavLLM for both Dense and Sparse Tasks

	ASR (WER% ↓)					ST (BLEU ↑)				
Full Token Baseline	2.25					21.56				
FLOPs Reduce	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
Weighted Merge	**2.25**	**2.44**	3.25	10.51	93.14	20.94	20.03	18.41	14.45	8.74
Weighted Merge + Constant Schedule	2.27	2.49	**2.48**	**2.96**	**4.73**	**21.47**	**20.72**	**19.81**	**18.54**	**17.45**
Weighted Merge + Decay Schedule	2.27	2.57	2.74	3.53	6.09	20.92	20.59	19.66	18.06	16.40

Table 8: The effectiveness of scheduler on WavLLM Dense tasks (ASR and ST)

4.3 Ablation Study

Effectiveness of Layer Selection: We analyze the effectiveness of our TE-based layer selection method in Table 6 as an ablation study. Several operation layers before layer 15 were selected to analyze the relationship between the TE and their actual performance. The results indicate that selecting the operational layer based on the TE rank (layer 3) can achieve the best performance on the ER task at most of the time. While the rank of TE may not be strictly proportional to the actual performance, in our study, TE serves as a theoretical reference for layer selection. A more comprehensive study on layer selection for token reduction is left for future research.

Effectiveness of Weighted Merge: Table 7 clearly illustrates the effectiveness of the weighted merge method. Compared to the normal average merge used in ToMe (Bolya et al., 2023) and A-ToMe (Li et al., 2023), our weighted merge algorithm consistently improves both ASR and ER in all the 10% to 50% FLOPs reduction ratio.

Effectiveness of Scheduling: For the dense tasks ASR and ST, we utilize the decay or constant scheduler to smoothly merge audio tokens which can prevent aggressive token dropping. As shown in Table 8, layer scheduler can greatly improve the performance of the dense task when the token reduction rate is very high. However, due to multiple operations across many layers, the pre-filling latency will increase. Therefore, a more careful design of the overall strategies is needed in the future to better manage the trade-off between performance and efficiency.

5 Conclusion

In this study, we propose **FastAdaSP**, an efficient inference framework that incorporates multiple stages in SpeechLMs. This preliminary study explores token reduction methods for SpeechLMs. We investigated various properties of different types of SpeechLM tasks and proposed novel methods for both dense and sparse tasks. Our method achieved a **1.84x** throughput increase with **7x** memory efficiency, setting a new benchmark for the efficiency-performance trade-off across various tasks.

Acknowledgments

Experiments of this work used the Bridges2 system at PSC and Delta system at NCSA through allocations CIS210014 and IRI120008P from the Advanced Cyber infrastructure Coordination Ecosystem: Services & Support (ACCESS) program, supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. 2023. Audiolm: a language modeling approach to audio generation. *Preprint*, arXiv:2209.03143.

Maxime Burchi and Valentin Vielzeuf. 2021. Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42:335–359.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. *Preprint*, arXiv:2403.06764.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. 2022. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518.

Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. 2023. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *Preprint*, arXiv:2311.07919.

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. MuTual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1406–1416, Online. Association for Computational Linguistics.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.

Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. 2024. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *Preprint*, arXiv:2402.09398.

Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. 2020. Clotho: An audio captioning dataset. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 736–740. IEEE.

Paul K. Rubenstein et al. 2023. Audiopalm: A large language model that can speak and listen. *Preprint*, arXiv:2306.12925.

Yuan Gong, Hongyin Luo, Alexander H. Liu, Leonid Karlinsky, and James R. Glass. 2024. Listen, think, and understand. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proc. INTERSPEECH 2020*, pages 5036–5040.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:3451–3460.

Shujie Hu, Long Zhou, Shujie Liu, Sanyuan Chen, Hongkun Hao, Jing Pan, Xunying Liu, Jinyu Li, Sunit Sivasankaran, Linquan Liu, and Furu Wei. 2024. Wavllm: Towards robust and adaptive speech large language model. *Preprint*, arXiv:2404.00656.

Kwangyoun Kim, Felix Wu, Yifan Peng, Jing Pan, Prashant Sridhar, Kyu J. Han, and Shinji Watanabe. 2023. E-branchformer: Branchformer with enhanced merging for speech recognition. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 84–91.

Sehoon Kim, Amir Gholami, Albert Shaw, Nicholas Lee, Karttikeya Mangalam, Jitendra Malik, Michael W Mahoney, and Kurt Keutzer. 2022. Squeezeformer: An efficient transformer for automatic speech recognition. In *Advances in Neural Information Processing Systems*, volume 35, pages 9361–9373. Curran Associates, Inc.

Yuang Li, Yu Wu, Jinyu Li, and Shujie Liu. 2023. Accelerating Transducers through Adjacent Token Merging. In *Proc. INTERSPEECH 2023*, pages 1379–1383.

Sihao Lin, Pumeng Lyu, Dongrui Liu, Tao Tang, Xiaodan Liang, Andy Song, and Xiaojun Chang. 2024. Mlp can be a good transformer learner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19489–19498.

Yichen Lu, Jiaqi Song, Xuankai Chang, Hengwei Bian, Soumi Maiti, and Shinji Watanabe. 2024. Syneslm: A unified approach for audio-visual speech recognition and translation via language model and synthetic data. *Preprint*, arXiv:2408.00624.

Soumi Maiti, Yifan Peng, Shukjae Choi, Jee-Weon Jung, Xuankai Chang, and Shinji Watanabe. 2024. Voxtlm: Unified decoder-only models for consolidating speech recognition, synthesis and speech, text continuation tasks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13326–13330.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. MELD: A multimodal multi-party dataset for emotion recognition in conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 527–536, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR.

Srijith Radhakrishnan, Chao-Han Yang, Sumeer Khan, Rohit Kumar, Narsis Kiani, David Gomez-Cabrero, and Jesper Tegnér. 2023. Whispering llama: A cross-modal generative error correction framework for speech recognition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10007–10016.

Ramon Sanabria, Ozan Caglayan, Shruti Palaskar, Desmond Elliott, Loïc Barrault, Lucia Specia, and Florian Metze. 2018. How2: a large-scale dataset for multimodal language understanding. In *Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL)*. NeurIPS.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. 2024. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *Preprint*, arXiv:2403.15388.

Justin Sirignano and Konstantinos Spiliopoulos. 2020. Mean field analysis of neural networks: A law of large numbers. *SIAM Journal on Applied Mathematics*, 80(2):725–752.

Guangzhi Sun, Wenyi Yu, Changli Tang, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun MA, Yuxuan Wang, and Chao Zhang. 2024a. video-SALMONN: Speech-enhanced audio-visual large language models. In *Forty-first International Conference on Machine Learning (ICML)*.

Zhenhong Sun, Ce Ge, Junyan Wang, Ming Lin, Hesen Chen, Hao Li, and Xiuyu Sun. 2022. Entropy-driven mixed-precision quantization for deep network design. In *Advances in Neural Information Processing Systems*, volume 35, pages 21508–21520. Curran Associates, Inc.

Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Daniel Cox, Yiming Yang, and Chuang Gan. 2024b. SALMON: Self-alignment with instructable reward models. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. 2024. Look-m: Look-once optimization in kv cache for efficient multimodal long-context inference. *Preprint*, arXiv:2406.18139.

Changhan Wang, Anne Wu, and Juan Pino. 2020. Covost 2 and massively multilingual speech-to-text translation. *Preprint*, arXiv:2007.10310.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. 2020. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7829–7833.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wenjing Zhu, Sining Sun, Changhao Shan, Peng Fan, and Qing Yang. 2024. Skipformer: A skip-and-recover strategy for efficient speech recognition. *Preprint*, arXiv:2403.08258.

A Appendix

A.1 Full Experiments Results

We also conduct the performance experiments on Qwen-Audio for both dense and sparse tasks and compare the baseline methods with our method. For the dense tasks ASR and ST, the results are presented in Table 9, demonstrating the effectiveness of our scheduling weighted token merge methods on another SpeechLM. The results for the sparse tasks ER and AC are shown in Table 10, which suggest our sparse setting method also performs well. These results on Qwen-Audio shows the effectiveness and generalization of our method across different SpeechLM.

Additionally, for the computation cost experiment, we also evaluated the Speech Summarization task on WavLLM using a subset of the How2 test set (Sanabria et al., 2018). As shown in Table 11, our method can effectively reduce the computation cost on a real dataset.

Further, we use one A100 80G GPU and one H100 80G GPU to conduct the long sequence experiments, which is shown in Table 12 and Table 13. The results indicate that increasing the audio length and beam size makes the acceleration of our method more noticeable.

A.2 Computation Reduction Theoretical Analysis

To analyze the computation reduction effect of our method, we use the theoretical FLOPs reduction rate. For simplicity, we just analysis the effective theoretical FLOPs reduction based on the token reduction rate and input sequence length on one layer. In the real situation, we can use the same methods to analyse all the decoder layers. Given the input sequence length n, the hidden dimension d and the Feed Forward Layer hidden dimension m. We can define the theoretical FLOPs in one transformer decoder layer as:

$$\text{FLOPs} = 2n^2d + 4nd^2 + 2ndm. \quad (10)$$

Where the first term represents the attention operation in equation 2; The second term represents the calculation of query, key, value and output tensors; The third term represents the calculation of the operation in Feed Forward Layer. Given the reduction ratio k, after the token reduction, we obtain the reduced sequence length $\hat{n} = n(1-k)$. Then the theoretical FLOPs reduction rate at the next layer can be calculated as:

$$\begin{aligned}\text{Rate} &= 1 - \frac{2\hat{n}^2d + 4\hat{n}d^2 + 2\hat{n}dm}{2n^2d + 4nd^2 + 2ndm} \\ &= 1 - \frac{2(1-k)^2n^2d + nd(1-k)(4d+2m)}{2n^2d + nd(4d+2m)} \\ &= k + \frac{(k-k^2)}{1 + \frac{(2d+m)}{n}} \propto n.\end{aligned}$$

As a result, the longer the input sequence length, the higher the FLOPs reduction rate that can be achieved. As demonstrated in A.1 long sequence speed test, the acceleration is more pronounced for a 240-second audio sample compared to a 120-second audio sample.

This theoretical computation cost analysis suggests that our method will result in greater computational reduction for longer audio sequence input, highlighting the effectiveness of this technique in real world applications where the input audio is often very long.

Algorithm 1 Weighted Token Merge Algorithm

procedure FASTADASP($A \in \mathbb{R}^{L\times D}, M \in \mathbb{R}^{T}, W_{\text{merge}} \in \mathbb{R}^{L}$)
  $i \leftarrow 1$ ▷ Index
  $H \leftarrow [\,]$ ▷ New hidden states
  while $i \leq L$ **do**
    $S \leftarrow \varnothing$ ▷ Initialize merge cluster
    $\mathbf{h} \leftarrow \omega_i \mathbf{a}_i$
    $t \leftarrow \omega_i$

    # Form the merge cluster
    while $i \in M$ **do**
      $S \leftarrow S \cup \{i\}$
      $i \leftarrow i + 1$
    end while

    # Perform Weighted Sum in Cluster
    for j **in** S **do**
      $\mathbf{h} \leftarrow \mathbf{h} + \omega_j \mathbf{a}_j$
      $t \leftarrow t + \omega_j$
    end for
    $\mathbf{h} \leftarrow \mathbf{h} \,/\, t$
    $H \leftarrow \text{append}(H, \mathbf{h})$
    $i \leftarrow i + 1$
  end while
  Output: $H \in \mathbb{R}^{N\times D}$
end procedure

	ASR (WER% ↓)					ST (BLEU ↑)				
Full Token Baseline	2.21					41.46				
FLOPs Reduce	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
Random Merge	2.43	3.39	8.21	27.53	169.96	40.63	39.35	37.01	32.39	24.3
Random Evict	5.70	21.42	61.04	184.59	342.88	38.39	28.22	14.98	6.29	-
A-ToMe (Li et al., 2023)	2.20	3.26	13.91	71.56	273.49	41.24	39.87	36.52	25.35	8.64
FastV (Chen et al., 2024)	12.54	54.40	110.42	179.58	258.78	41.12	40.31	38.45	34.74	27.14
FastAdaSP-Dense Decay Schedule	**2.19**	2.23	2.51	4.37	**15.24**	41.41	41.05	40.51	39.02	35.79
FastAdaSP-Dense Constant Schedule	2.22	**2.21**	**2.30**	**3.57**	16.01	**41.47**	**41.30**	**40.83**	**39.81**	**37.04**

Table 9: Comparison between FastAdaSP with other token reduction methods on Qwen-Audio **dense tasks**

	ER (ACC% ↑)					AC (CIDEr ↑ | SPICE ↑ | SPIDEr ↑)				
Full Token Baseline	54.80					0.45 | 0.13 | 0.29				
FLOPs Reduce	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
Random Merge	51.80	48.00	43.80	39.20	32.30	0.44 | 0.13 | 0.29	0.43 | 0.13 | 0.28	0.41 | 0.13 | 0.27	0.41 | 0.12 | 0.26	0.38 | 0.12 | 0.25
Random Evict	52.80	48.20	42.00	34.61	23.14	0.43 | 0.13 | 0.28	0.42 | 0.13 | 0.27	0.38 | 0.12 | 0.25	0.31 | 0.10 | 0.20	0.12 | 0.07 | 0.14
A-ToMe (Li et al., 2023)	54.91	54.70	54.20	**53.90**	51.60	0.44 | 0.13 | 0.29	0.44 | 0.13 | 0.29	0.43 | 0.13 | 0.28	0.41 | 0.13 | 0.27	0.39 | 0.12 | 0.28
FastV (Chen et al., 2024)	54.80	53.80	53.50	52.10	50.38	0.44 | 0.13 | 0.29	**0.45 | 0.13 | 0.29**	0.45 | 0.13 | 0.29	0.44 | 0.13 | 0.28	0.43 | 0.13 | 0.28
FastAdaSP-Sparse	**55.17**	**55.05**	**54.40**	53.86	**52.14**	**0.45 | 0.13 | 0.29**	0.44 | 0.13 | 0.29	**0.45 | 0.13 | 0.29**	**0.44 | 0.13 | 0.28**	**0.43 | 0.13 | 0.28**

Table 10: Comparison between FastAdaSP with other token reduction methods on Qwen-Audio **sparse task**

A.3 FastAdaSP: Algorithm Details

Here we show the full implementation details of the **FastAdaSP** algorithm, which was brifely mentioned in Section 3.2. Given the audio feature sequence $A = (\mathbf{a}_i \in \mathbb{R}^D | i = 1, ..., L)$, the merge index list $M = (m_i \in \mathbb{R} | i = 1, ..., T)$ and merge weights $W_{\text{merge}} = (\omega_i \in \mathbb{R} | i = 1, ..., L)$. Then we can use Algorithm 1 to obtain the merged audio feature sequence $H = (\mathbf{h}_i \in \mathbb{R}^D | i = 1, ..., N)$, where N is the length of the merged audio feature sequence.

Additionally, if there are B batches in the hidden states, we currently need to perform the algorithm B times to reduce the audio tokens for each audio sequence separately. In the future, this process may be improved by executing the algorithm for each batch in parallel.

A.4 Derivation of Transfer Entropy

In this section, we recall the derivation of transfer entropy from (Lin et al., 2024). We also did a slight modification on the final definition based on our settings. As mentioned in section 3.4, given $F \in \mathbb{R}^{L \times D}$ as the feature output after attention block, the entropy was defined as:

$$H(F) = -\int p(f) \log p(f)\, df, \; f \in F. \quad (11)$$

Following the (Lin et al., 2024; Sirignano and Spiliopoulos, 2020), we regard the feature F's probability distribution as a Gaussian distribution $F \sim \mathcal{N}(\mu, \sigma^2)$. Therefore, the equation 11 can be derived into:

$$\begin{aligned} H(F) &= -\mathbb{E}[\log \mathcal{N}(\mu, \sigma^2)] \\ &= -\mathbb{E}\left[\log\left[(2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(f-\mu)^2\right)\right]\right] \\ &= \log(\sigma) + \frac{1}{2}\log(2\pi) + \frac{1}{2} \end{aligned}$$

Where σ_i is the standard deviation of i-th hidden state in F. The $H(F)$ is proportional to the $\log(\sigma)$ since $\frac{1}{2}\log(2\pi) + \frac{1}{2}$ is constant term. Thus we could get the equation 7 in Sec 3.4.

A.5 Applications in the Real World and Future Perspective

In this study, we propose a efficient inference framework which designed for audio modality reduction in Multitask SpeechLM. In the context of long audio sequences, it is observed that only a small part of tokens carries critical information, while others may be not relevant (e.g. periods of noisy or blank audio). Our proposed plug-and-play methodology aims to efficiently identify and prioritize significant audio tokens during the pre-filling

Token Reduce %	SUM (ROUGE-L ↑)	FLOPs Reduction % ↑	Real Time Factor ↓	Pre-filling Latency (s) ↓	Decoding Latency (s) ↓	Throughput (token/s) ↑
Full Token Baseline	16.20	0.00	0.091	0.51	5.09	15.57
10	16.63	9.54	0.090	0.51	4.92	16.00
20	16.27	19.05	0.087	0.51	4.74	16.02
30	16.29	28.46	0.083	0.51	4.52	16.05
40	15.29	37.66	0.083	0.51	4.51	16.62
50	15.10	46.97	0.078	0.51	4.20	16.87

Table 11: **Computational cost experiments on Real Dataset.** Inference result of 100 How2 test set samples around 60s on WavLLM using one V100 32GB GPU

Beam Size	Audio Length (s)	Token Reduce %	FLOPs Reduction % ↑	Real Time Factor ↓	Pre-filling Latency (s) ↓	Decoding Latency (s) ↓	Throughput (token/s) ↑
1	120	Full Token	0.00	0.054	0.79	5.75	12.86
		50	48.62	0.044	0.77	4.57	13.57 (1.05x)
5	120	Full Token	0.00	0.137	3.11	13.32	5.48
		50	48.40	0.092	3.09	8.01	8.87 (1.61x)
1	240	Full Token	0.00	0.044	1.70	8.90	8.09
		50	49.21	0.036	1.59	7.02	9.69 (1.20x)
5	240	Full Token	0.00	0.126	6.72	23.55	3.10
		50	49.21	0.077	6.48	11.89	5.72 (**1.84x**)

Table 12: **Long Sequence Computational cost experiments on A100.** Long sequence audio samples (120s and 240s) input on WavLLM using one A100 80GB GPU

Beam Size	Audio Length (s)	Token Reduce %	FLOPs Reduction % ↑	Real Time Factor ↓	Pre-filling Latency (s) ↓	Decoding Latency (s) ↓	Throughput (token/s) ↑
1	120	Full Token	0.00	0.027	0.26	3.00	24.63
		50	48.62	0.023	0.26	2.52	24.73 (1.01x)
5	120	Full Token	0.00	0.043	0.48	4.73	15.44
		50	48.40	0.032	0.46	3.44	20.66 (1.34x)
1	240	Full Token	0.00	0.020	0.43	4.29	16.70
		50	49.21	0.019	0.39	4.06	16.75 (1.00x)
5	240	Full Token	0.00	0.039	1.13	8.39	8.70
		50	49.21	0.026	0.96	5.42	12.55 (**1.44x**)

Table 13: **Long Sequence Computational cost experiments on H100.** Long sequence audio samples (120s and 240s) input on WavLLM using one H100 80GB GPU

stage, which can offers substantial benefits for long-form audio comprehension.

In addition, in practical deployments of SpeechLM products, batch decoding is often a necessity, with batch sizes potentially reaching up to 128 or more. Within these batch decoding settings, our proposed methods are designed to reduce the memory footprint associated with many long audio inputs while simultaneously accelerating the decoding process. This optimization is crucial for enhancing the efficiency and scalability of SpeechLM systems in real world applications.

In the future, we may extend the current efficient inference framework to multi-round decoding scenarios, which can handle the dense task and sparse task at the same time. This improvement will make the whole system more applicable to real world use cases. Moving forward, this pioneering study on audio token reduction techniques in Multimodal Large Language Models (MLLM) paves the way for future research to explore the general behavior of audio and other modalities such as vision. The next stage of this study is to investigate the unified methodology to accelerate both audio and vision modalities simultaneously in Audio-Visual LLMs (e.g., video-SALMONN (Sun et al., 2024a)), which enable more efficient inference for long video understanding.