Enhancing Speech Large Language Models with Prompt-Aware Mixture of Audio Encoders

Anonymous ACL submission

Abstract

Connecting audio encoders with large language models (LLMs) allows the LLM to perform various audio understanding tasks, such as automatic speech recognition (ASR) and audio captioning (AC). Most research focuses on training an adapter layer to generate a unified audio feature for the LLM. However, different tasks may require distinct features that emphasize either semantic or acoustic aspects, making task-specific audio features more desirable. In this paper, we propose Prompt-aware Mixture (PaM) to enhance the Speech LLM that uses multiple audio encoders. Our approach involves using different experts to extract different features based on the prompt that indicates different tasks. Experiments demonstrate that with PaM, only one Speech LLM surpasses the best performances achieved by all single-encoder Speech LLMs on ASR, Speaker Number Verification, and AC tasks. PaM also outperforms other feature fusion baselines, such as concatenation and averaging.

1 Introduction

Large language models (LLMs) have demonstrated exceptional performance across various natural language processing tasks, including question answering, machine translation, and summarization (OpenAI, 2023). In recent work, there has been a growing focus on merging speech encoders with LLMs, so that the LLM can understand the spoken content without the need for explicit transcription, promoting tasks such as direct speech translation (Chen et al., 2024) and named entity recognition from speech (Li et al., 2024). Much of this work leverages adapter layers to downsample and map speech features into the LLM's embedding space. These adapter layers include various forms, such as attention layers (Yu et al., 2024), adaptive CTC downsamplers (Ling et al., 2023), and convolutional layers (Fathullah et al., 2023). Beyond semantic understanding tasks, Speech LLMs have been extended to encompass a broader range of applications, including audio event detection and audio captioning (Chu et al., 2024).

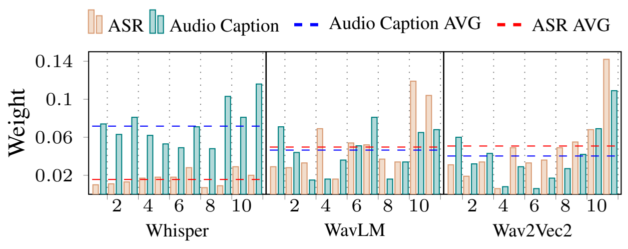

Figure 1: ASR and Audio Caption tasks favor different encoders and layers of features. The dotted lines indicate the average (AVG) importance of different encoders. The bar chart represents fine-grained layer importance.

Multitasking requires that the input audio features contain as much relevant information as possible, representing the input speech, which may include speech content, noise, and speaker-specific characteristics. When fine-tuning self-supervised speech encoders, researchers assign learnable weights to each layer and observe that different downstream tasks prioritize different levels of features (Chen et al., 2022). In our Speech LLM framework, a similar trend is evident, where different tasks prioritize different encoders and feature levels (Figure 1). These biases arise from the inherent differences in the tasks themselves. For instance, the automatic speech recognition (ASR) task focuses solely on the speech content, disregarding other factors such as speaker characteristics and background noise. In contrast, tasks like audio captioning (AC) may rely on these additional factors that ASR intentionally excludes.

Consequently, researchers have proposed using multiple encoders to extract more robust features. For instance, WavLLM (Hu et al., 2024) employs both the WavLM (Chen et al., 2022) and the Whisper (Radford et al., 2022) encoder, while SALMONN (Tang et al., 2024) integrates the Whisper encoder and the BEATs (Chen et al., 2023). However, these approaches consider all encoders equally important, and merge the features from different encoders based on simple concatenation method across all tasks. Moreover, MoWE (Zhang et al., 2024) employs a strong encoder and multiple weaker encoders via the Mixture of experts (MoE) approach. However, MoWE only utilizes the input audio to control the gating mechanism, without incorporating task-specific information in prompts.

In this paper, we introduce Prompt-aware Mixture (PaM), a novel MoE method for merging multiple encoders to enhance Speech LLMs, our approach integrates a prompt-aware gating mechanism, emphasizes feature fusion, and considers the relative importance of each encoder for different tasks, aiming to improve all downstream performance. PaM employs three distinct audio encoders: the Whisper encoder (Radford et al., 2022), WavLM (Chen et al., 2022), and Wav2Vec2 (Baevski et al., 2020). We train a set of experts for prompt-aware feature fusion, comprising one shared expert and four task-specific experts. On each task, an expert learns the optimal weights for each encoder and its respective layers, and subsequently maps the resulting features to the embedding space of the Qwen2.5 model (Team, 2024). The embedding of the prompt is utilized to determine the appropriate routing. Notably, in PaM, the routing guides the selection of the fusion parameters rather than the choice of the encoder. Experiments are conducted across three tasks: ASR, speaker number verification, and AC. On all datasets, including LibriSpeech (Panayotov et al., 2015), AMI (Kraaij et al., 2005), AIR-Bench (speaker number verification) (Yang et al., 2024), and AudioCaps (Kim et al., 2019), PaM consistently enhances performance in comparison to the best-performing single-encoder Speech LLM. The relative improvements are 15%, 25%, 3.4%, and 7.6%, respectively, based on the corresponding evaluation metrics. Additionally, we observed that without PaM, simple concatenation and averaging are generally effective for AC tasks; however, they result in performance declines for ASR. Our contributions can be summarized as follows:

- We propose a novel multi-encoder Speech LLM, which effectively leverages features from every layer of each encoder.
- We introduce PaM, a specialized MoE method that incorporates a prompt-aware gating mechanism to assign distinct weights to each encoder and their layers based on the task.
- We conducted comprehensive experiments demonstrating that PaM significantly enhances the overall performance of all downstream tasks. Additionally, we present detailed feature importance analyses and explore various combinations of speech encoders and LLMs.

2 Method

In this section, we begin with an overview of the proposed PaM method (Figure 2). We then elaborate on the details of the encoder fusion process, executed by a single expert, and describe the prompt-aware routing method.

2.1 Overall Architecture

The architecture of the proposed PaM method is depicted in Figure 2. As described in Equation 1, the LLM accepts the text prompt $\mathbf{X}_{\text{prompt}}$, which includes task-related information, along with the speech features $\mathbf{H}_{\text{audio}}$ as input, and subsequently generates the response $\mathbf{Y}$.

$$\mathbf{Y} = \text{LLM}(\mathbf{X}_{\text{prompt}}, \mathbf{H}_{\text{audio}}) \tag{1}$$

To obtain $\mathbf{H}_{\text{audio}}$, we employ three encoders: the Whisper encoder, WavLM, and Wav2Vec2. For each encoder, the hidden states are initially processed by a feed-forward network (FFN) as described in Equation 2, resulting in the feature of each encoder, denoted as $\mathbf{H}_i$.

$$\mathbf{H}_i = \text{FFN}_i(\text{Encoder}_i(\mathbf{X}_{\text{audio}})) \tag{2}$$

Next, as shown in Equation 3, we combine these features using an MoE fusion method, which includes a shared expert and N routed experts where the number of routed experts corresponds to the number of predefined tasks. During inference, only one routed expert is selected, based on the task indicated by the prompt.

$$\begin{aligned}\mathbf{H}_{\text{audio}} &= \text{Expert}_{\text{share}}(\mathbf{H}_{\{1,2,3\}}) \\ &+ \sum_{j=1}^{N} G_j(\mathbf{X}_{\text{prompt}}) \times \text{Expert}_j(\mathbf{H}_{\{1,2,3\}}) \end{aligned} \tag{3}$$

Overall, each expert processes the features from all encoders ($\mathbf{H}_{\{1,2,3\}}$) for feature fusion. The shared expert extracts common features for all tasks, while the routed expert performs task-specific feature fusion. The routing is determined by the user input, which is the prompt.

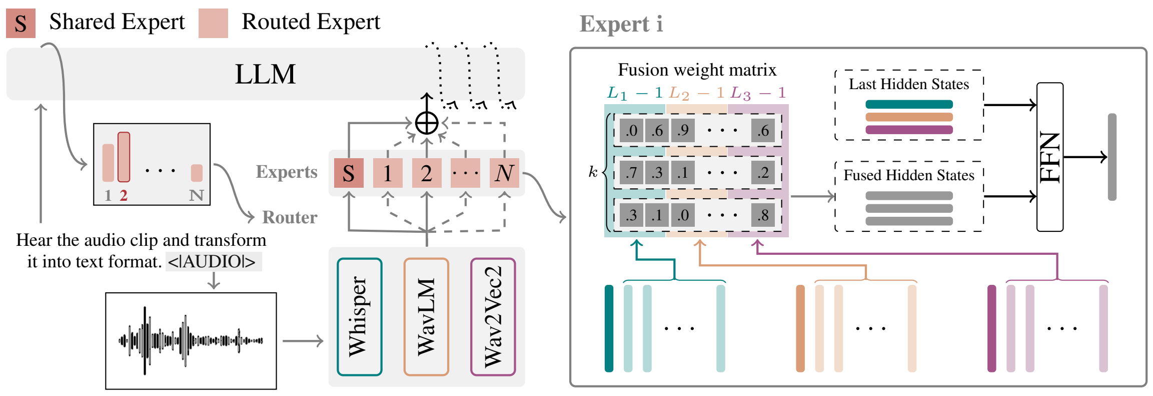

Figure 2: The architecture of the proposed PaM method. The output feature of the prompt guides the routing of the MoE adapter where we use a fixed shared expert and a single routed expert. For each expert, the last hidden states of all encoders are concatenated with a set of fused hidden states derived from a fusion weight matrix. Subsequently, an FFN is applied to align with the dimensions of the LLM.

2.2 Multi-layer Fusion

Different encoders exhibit distinct strengths. For example, WavLM is excellent at extracting speaker information, while Wav2Vec2 excels in capturing semantic content. The Whisper encoder, trained on a vast amount of data, provides superior features for AC and ASR in noisy environments. Additionally, features from different layers contain varying levels of information. Deeper layers hold high-level semantic information, whereas lower layers may contain fine-grained acoustic details. Thus, for feature fusion, we consider features from all layers of all three encoders. Specifically, for the feature $\mathbf{H}_i$ from a single encoder, it includes hidden states from all L_i Transformer (Vaswani et al., 2017) layers as well as $\mathbf{h}_i^0$, the hidden states following the convolutional layers (Equation 4).

$$\mathbf{H}_i = \{\mathbf{h}_i^0, \mathbf{h}_i^1, ...\mathbf{h}_i^{L_i}\} \tag{4}$$

As illustrated in Equation 5, we consider the hidden states $\mathbf{h}_i^{\{0\text{-}(L_i\text{-}1)\}}$ to derive the fused hidden states $\mathbf{h}_k^{\text{fused}}$. For the hidden states of each layer, a single scalar weight is assigned to control its importance. We utilize a set of scalar weight include k elements $\{w_{i,j}^1, \ldots, w_{i,j}^k\}$ for each hidden state $\mathbf{h}_{i,j}$ from all three encoders to generate three fused hidden states $\mathbf{h}_{\{1,\ldots,k\}}^{\text{fused}}$. Thus, the dimension of the whole matrices is $\mathbf{W} \in \mathbb{R}^{3\times(L_1+L_2+L_3)}$.

$$\mathbf{h}_k^{\text{fused}} = \sum_{i=1}^{3}\sum_{j=1}^{L_i} w_{i,j}^k \cdot \mathbf{h}_{i,j} \tag{5}$$

Finally, we concatenate the last hidden states of the three encoders $\mathbf{h}_{\{1,2,3\}}^{L_i}$ with the three fused hidden states $\mathbf{h}_{\{1,\ldots,k\}}^{\text{fused}}$ along the feature dimension. Afterward, we apply a FFN to compress the feature dimension to match the dimension of the LLM embedding (Equation 6).

$$\begin{aligned}\mathbf{h}^{\text{final}} &= \text{Concat}(\mathbf{h}_{\{1,2,3\}}^{L_i}, \mathbf{h}_{\{1,\ldots,k\}}^{\text{fused}}) \\ \text{Expert}(\cdot) &= \text{FFN}(\mathbf{h}^{\text{final}})\end{aligned} \tag{6}$$

The parameters in our fusion method are independent among the routed experts. The use of fusion weight matrices highlights multi-level feature fusion, while the final concatenation followed by the FFN provides more fine-grained feature fusion.

2.3 Prompt Aware Routing

A prompt refers to a segment of text that provides context or objectives for the generation of the Speech LLM. Although the format of prompts can vary widely, they can typically be categorized into several distinct types according to the task the user wishes to perform. For instance, speech-related tasks, sound-related tasks and speech chat tasks (Yang et al., 2024). In this paper, we investigate three tasks: ASR, speaker number verification, and AC. We utilize distinct experts for each task. For effective routing, the router must identify the task type based on the prompt. We employ a simple classification approach (Equation 7 and 8) wherein we use the last hidden states $\mathbf{H}_{\text{prompt}}$ of the prompt from the LLM, followed by a FFN and Softmax activation, to obtain the task posteriors $\text{P}(\text{Task}|\mathbf{H}_{\text{prompt}})$.

$$\mathbf{H}_{\text{prompt}} = \text{LLM}(\mathbf{X}_{\text{prompt}}) \tag{7}$$

$$\text{P}(\text{Task}|\mathbf{H}_{\text{prompt}}) = \text{Softmax}(\text{FFN}(\mathbf{H}_{\text{prompt}})) \tag{8}$$

ASR	1. Hear the audio clip and transform it into text format. <|AUDIO|> 2. Listen to the following audio and create a corresponding text transcript. <|AUDIO|>
Speaker Number Verification	1. How many speakers' contributions are in this recording? <|AUDIO|> 2. What is the number of speakers in this spoken content? <|AUDIO|>
AC	1. Listen to this audio and provide a detailed description. <|AUDIO|> 2. Analyze the recording and summarize its contents. <|AUDIO|>

Table 1: Examples of prompts for different tasks.

As shown in Equation 9, we select the routed expert with the Top-1 probability by the indicator function.

$$\mathrm{G}_j = \begin{cases} 1 & \text{if } j \in \text{Top-1}(\mathrm{P}(\text{Task}|\mathbf{H}_{\text{prompt}})) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

To train the FFN, we create diverse prompts for each task using the LLM. Specifically, we manually write several prompts for each task and instruct ChatGPT to rewrite these prompts. Examples of these prompts are shown in Table 1. It is important to note that the audio features follow the prompt because we use the prompt to guide feature extraction and fusion. This approach differs from other works, where the audio features <|AUDIO|> can be positioned before the prompt.

2.4 Training Objective

The training loss function, as illustrated in Equation 10, is the sum of the cross-entropy loss $\mathcal{L}_{\mathrm{G}}$ for prompt-aware gating and the cross-entropy loss $\mathcal{L}_{\text{llm}}$ between the LLM's output $\mathbf{Y}$ and the ground truth $\hat{\mathbf{Y}}$.

$$\begin{aligned} \mathcal{L} = \mathcal{L}_{\mathrm{G}}(\mathrm{P}(\text{Task}|\mathbf{H}_{\text{prompt}}), \text{Task}) \\ + \mathcal{L}_{\text{llm}}(\mathbf{Y}, \hat{\mathbf{Y}}) \end{aligned} \tag{10}$$

3 Experimental Setups

In this section, we present a detailed description of our experimental setups, covering datasets, evaluation metrics, hyperparameters for the model architecture, and training. The links to the pretrained models and datasets used can be found in Appendix A.1, while the implementation details of baseline methods are available in Appendix A.2.

3.1 Datasets and Evaluation Metrics

We assess the efficacy of our method across three audio-to-text tasks: automatic speech recognition (ASR), speaker number verification (SNV), and audio captioning (AC). The training set includes the following: 200 hours of ASR data from LibriSpeech-100 (Panayotov et al., 2015) and AMI (Kraaij et al., 2005), 150 hours of SNV data synthesized from Common Voice V4 (Ardila et al., 2019), and 100 hours of AC data from AudioCaps (Kim et al., 2019). In total, the training data contains 450 hours of audio signals. For SNV, we randomly concatenate individual utterances to form new speech signals with the number of speakers ranging from one to four. The test dataset includes LibriSpeech-test-clean, LibriSpeech-test-other, AMI, the SNV test set from AIR-Bench (Yang et al., 2024), and the test set of AudioCaps along with its corresponding QA version from AudioBench (Wang et al., 2024), which contains diverse questions. ASR tasks focus on semantics. The LibriSpeech test set originates from audiobooks, demonstrating ASR performance in a clean scenario. AMI, a real meeting corpus containing spontaneous talk, reflects ASR performance in a more challenging, real-world scenario. SNV and AC test sets can indicate the Speech LLM's ability to understand speaker and acoustic information. We evaluate the performance using word error rate (WER) for ASR tasks, accuracy for SNV, and METEOR (Banerjee and Lavie, 2005) for AC.

3.2 Model Architecture and Training

We train our model based on Huggingface Transformers Library[1]. Our model consists of three audio encoders, a pre-fusion adapter for each encoder, a PaM fusion module, and an LLM. In our main experiments, the encoders are Whisper-Small (Radford et al., 2022), WavLM-Base-Plus (Chen et al., 2022), and Wav2Vec2-Base-960h (Baevski et al., 2020), each with approximately 100 million parameters. We downsample the features from the Whisper encoder by a factor of two, resulting in a frame length of 40ms, consistent with the frame length of Wav2Vec2 and WavLM. The pre-fusion adapter is an FFN that transforms the encoder's hidden dimension D_{E} to the LLM's hidden dimension D_{LLM}. Each expert in the PaM fusion module includes a fusion weight matrix ($\mathbb{R}^{3L\times 3}$) and a linear layer ($\mathbb{R}^{6D_{\text{LLM}}\times D_{\text{LLM}}}$) to fuse features from all

[1] https://github.com/huggingface/transformers

Model	LibriSpeech		AMI	SNV	AudioCaps	AudioCaps QA	AVG Rank↓
	WER(clean)↓	WER(other)↓	WER↓	Acc↑	METEOR↑	METEOR↑	
Single-encoder Baselines							
- Whisper (Radford et al., 2022)	9.61	16.73	**16.27**	18.80%	**32.96**	**15.04**	5.67
- WavLM (Chen et al., 2022)	5.59	10.57	18.97	**41.40%**	27.14	12.77	5.50
- Wav2Vec2 (Baevski et al., 2020)	**4.30**	**09.46**	26.69	39.00%	23.81	11.20	5.33
Multi-encoder Baselines							
- WavLLM (Hu et al., 2024)	4.95 (- 0.65)	09.19 (+0.27)	15.29 (+0.98)	39.20% (- 2.20)	34.93 (+1.97)	16.35 (+1.31)	2.67
- SALMONN (Tang et al., 2024)	5.04 (- 0.74)	09.70 (- 0.24)	19.04 (- 2.77)	49.40% (+8.00)	34.86 (+1.90)	15.97 (+0.93)	3.50
- Average	4.76 (- 0.46)	10.43 (- 0.97)	17.20 (- 0.93)	45.50% (+4.10)	33.22 (+0.26)	15.53 (+0.49)	3.67
PaM (Ours)	3.65 (+0.65)	07.07 (+2.39)	12.79 (+3.48)	42.80% (+1.40)	35.47 (+2.51)	15.70 (+0.66)	**1.67**

Table 2: Comparison of the proposed PaM method with single and multi-encoder baselines. For multi-encoder baselines: WavLLM and SALMONN use concatenation followed by a linear layer and Q-former, respectively, while 'Average' averages the three features. Values in the brackets indicate performance improvement (green) or degradation (red) compared to the best single encoder result. The avg rank column shows the average rank on each task. Smaller ranks indicate better performance.

encoders. Here, L represents the number of layers in the encoder, which is 12 for all encoders in our experiments. For the fused features, we set $k = 3$, corresponding to the number of last hidden states. We utilize four routed experts, each corresponding to a specific task category: ASR-clean, ASR-noisy, SNV, and AC. For each category, we generate 50 prompts using ChatGPT (OpenAI, 2023). For the LLM model, we select the Qwen2.5-3B (Team, 2024). In section 4, we also experiment with other encoders, including Hubert-Base-LS960, Whisper-Large-v3, and WavLM-Large.

We train our model for five epochs with a learning rate of 5e-5, 2000 warmup steps, and bf16 precision. We freeze all encoders and the LLM, training only the added parameters of the adapters and the fusion modules. For the LLM, we apply LoRA (Hu et al., 2022) with a rank of 32 and an alpha of 64, adding LoRA only on the q_proj and k_proj. Each task has the same probability during training. During the inference stage, we select the last checkpoint on the validation set and perform greedy search.

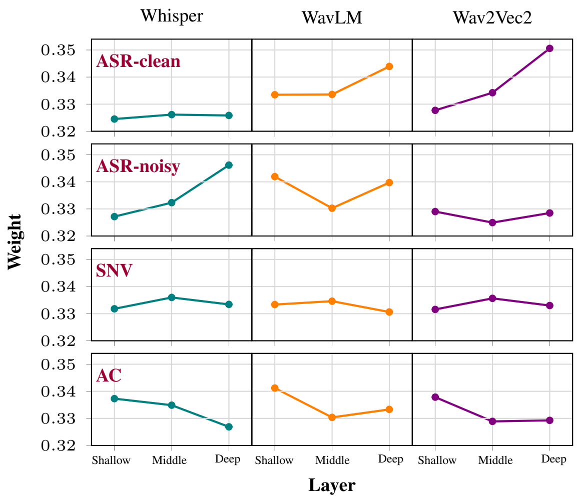

Figure 3: The weights for each encoder and its layers.

4 Results

4.1 Main Results

As demonstrated in Table 2, we compare the proposed PaM method with single and multi-encoder baselines. Each encoder exhibits distinct advantages. When utilizing a single encoder, the Speech LLM with the Whisper encoder performs best on the AMI dataset and AC tasks. The primary reason is that the Whisper model is trained on a vast amount of speech data, exposing it to diverse acoustic conditions. Consequently, it excels in challenging ASR and AC tasks in real-world environments and noisy conditions. The WavLM encoder, trained on multi-speaker speech signals, provides the best features for the SNV task. The Wav2Vec2 encoder performs best on the LibriSpeech dataset mainly because it was pretrained on this dataset. However, since the LibriSpeech dataset consists of clean audiobooks, the Speech LLM with the Wav2Vec2 encoder shows poor performance on the AMI and AudioCap datasets.

We reimplemented the feature fusion methods of WavLLM (Hu et al., 2024) and SALMONN (Tang et al., 2024), training the Speech LLM with the three audio encoders in our setups. Both methods use concatenation but are followed by different projection layers: WavLLM with a linear layer and SALMONN with a Q-former layer. Additionally, we implemented a simple averaging method that directly computes the average of the features from the three encoders. Compared to the best

Encoders	LibriSpeech		AMI	SNV	AudioCaps	AudioCaps QA	AVG Rank↓
	WER(clean)↓	WER(other)↓	WER↓	Acc↑	METEOR↑	METEOR↑	
Whisper+X	4.42	8.92	13.96	43.70%	**35.41**	16.11	4.5
WavLM+X	4.07	7.97	18.47	47.47%	32.48	15.01	5.5
Wav2Vec2+X	**3.42**	7.20	18.18	45.13%	32.33	15.02	4.3
HuBERT+X	3.87	8.21	19.49	36.90%	31.82	15.08	6.8
Whisper+X+Y	4.22	8.11	**13.79**	**49.77%**	35.37	**16.31**	**3.0**
WavLM+X+Y	3.72	7.99	16.35	38.73%	34.23	15.82	4.0
Wav2Vec2+X+Y	3.77	**6.90**	16.90	42.63%	34.23	15.82	3.8
HuBERT+X+Y	4.03	7.96	17.13	44.17%	34.18	16.13	4.0

Table 3: Results with different combinations of encoders. The first and last four rows represent combinations of two and three encoders respectively. Each row's results are the average performance of a fixed encoder paired with all possible combinations of one or two other encoders, highlighting the unique strengths of each encoder.

performance of single encoder baselines, all three fusion methods achieve better METEOR scores on AC tasks. However, performance may degrade on other tasks. For example, we observed performance degradation for all three methods on the LibriSpeech test-clean subset. This is expected since the same features are used for all tasks. Features containing more useful acoustic information for AC tasks may lack useful semantic information for ASR tasks.

PaM consistently outperforms all single encoder baselines, delivering performance improvements across all tasks. This consistent improvement can be attributed to the MoE adapter, which provides unique features tailored for each task. Compared to other fusion methods (i.e., concatenation and averaging), PaM achieves significantly lower WERs on the LibriSpeech and AMI datasets and similar performance on SNV and AC tasks.

4.2 Feature Importance

In Figure 3, we visualize the fusion weights for each expert, excluding the shared expert, which can be interpreted as the fusion weight for each task. To enhance clarity, we summed the weights for every four layers, resulting in the total weight for shallow, middle, and deep layers. Generally, different tasks require different features, so each expert has distinct fusion weights. Specifically, when the expert for ASR-clean is activated, it mainly focuses on features from WavLM and Wav2Vec2, especially the deep layers. When the expert for ASR-noise is activated, it primarily focuses on features from the Whisper encoder and WavLM. For SNV and AC tasks, all three encoders have similar fusion weights. For the SNV task, features from the middle layers are more important, while for the AC task, features from the shallow layers are more important.

4.3 Combinations of Encoders

In Table 3, we extend our investigation to encompass more combinations of encoders, including two and three encoders, and incorporate the HuBERT encoder. To highlight the strengths of each encoder, we calculate the performance by keeping one encoder fixed and varying the other encoders, then computing the average. The average (AVG) rank reflects the overall multitask performance. It is evident that using three encoders significantly outperforms using two encoders in all combinations, thereby demonstrating the effectiveness of employing more encoders for Speech LLMs.

We can observe that different encoders offer varying benefits, which proves that the tasks-specific encoders could significantly improve the performance on corresponding tasks. For example, when Whisper is used, regardless of whether two or three encoders are employed, the Speech LLM achieves the lowest WER on AMI and the highest METEOR scores on AudioCaps. On the other hand, Wav2Vec2 provides an advantage for recognizing speech signals in LibriSpeech. This indicates that when selecting encoders for Speech LLM, it is essential to consider the domain, downstream tasks, and the capabilities of each encoder. It is suggested to use a robust general domain model like Whisper in combination with domain-specific encoders such as Wav2Vec2.

4.4 Larger Encoders and LLMs

We try to further enhance performance by replacing the encoders in the proposed method with their larger versions (Table 4). Specifically, we replace the Whisper-Small encoder with the Whisper-Medium encoder, Wav2Vec2-Base-960h

Models	LibriSpeech		AMI	SNV	AudioCaps	AudioCaps QA
	WER(clean)↓	WER(other)↓	WER↓	Acc↑	METEOR↑	METEOR↑
Base PaM	3.65	07.07	12.79	42.8%	35.47	15.70
PaM with Larger Encoders						
- ① Whisper-Medium	3.93	07.93	12.43	47.5%	35.61	15.58
- ② WavLM-Large	3.75	06.43	12.10	45.6%	35.95	16.71
- ③ Wav2Vec2-Large	3.74	06.49	15.06	47.6%	35.50	**16.74**
- ① + ② + ③	**3.58**	**05.93**	**11.51**	**56.9%**	**36.94**	16.50
PaM with different LLMs						
- Qwen2.5-7B	3.68	08.36	15.26	43.7%	35.46	15.71
- LLaMA3.2-3B	4.98	11.57	15.57	50.5%	35.83	16.34
- LLaMA3.1-8B	4.85	08.87	15.01	50.8%	35.81	15.82

Table 4: Results with larger encoders and various LLMs. To enhance performance, we replaced the Base version's encoders and experimented with different LLMs.

with Wav2Vec2-Large-960h, and WavLM-Base-Plus with WavLM-Large. Our observations indicate that on the LibriSpeech-clean dataset, performance does not significantly improve and may even slightly degrade. However, for the SNV and AC tasks, performance consistently improves, suggesting that more challenging sound-related tasks benefit more from better encoders. Additionally, we observe that when all encoders are replaced with their larger versions, we achieve the best performance across almost all tasks, albeit at the cost of increased computation.

In our investigation of other LLMs, including Qwen2.5-7B, LLaMA3.2-3B, and LLaMA3.1-8B, we observed some improvements in certain tasks. However, the overall performance was not superior to that of Qwen2.5-3B. The potential reason for this is that we used short audios, and both the prompts and answers were brief, thereby not fully utilizing the strong semantic understanding capabilities of the larger LLMs. Consequently, we opted to use Qwen2.5-3B in this paper. It is important to emphasize that for Speech LLMs, the extracted features may be more critical than the LLM itself for many downstream tasks.

4.5 Parameters of the Adapter

In PaM, we employ multiple experts, merge and concatenate various features. Consequently, the number of parameters is slightly higher than that of the baseline Concatenation and Average methods. To ensure a fair comparison, we reduce the dimensionality within PaM, resulting in a parameter count of only 29M, which is similar to the baselines. As observed in Table 4, PaM outperforms or performs comparably to the baselines across all tasks.

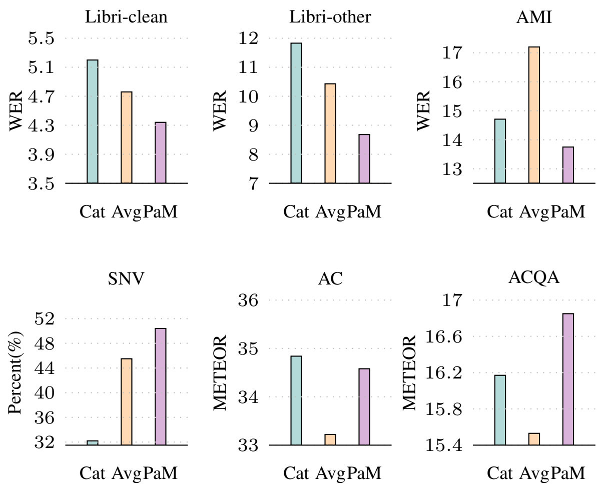

Figure 4: Performance comparison of a smaller PaM (29M parameters) with Concatenation (37M parameters) and Average (24M parameters).

5 Discussions

More and Unseen Tasks: Although our experiments involve three tasks and five datasets, they are representative as they encompass both semantic and acoustic-related tasks. PaM can be extended to other tasks, such as speech translation, which requires features similar to ASR and thus is not included in our paper. Additionally, PaM is applicable to unseen tasks and datasets. To achieve this, tasks must be categorized based on similar feature requirements, and routing during inference should be based on these categories.

Alignment: Although our paper primarily emphasizes the role of PaM in feature fusion, it also serves an alignment function by mapping the different fused features into the LLM's embedding space. Since we use different features for different prompts/tasks, the projections needed for feature aligment also vary. This is why each expert not only has fusion parameters but also its own FFN for projection.

6 Related Works

Audio Encoders: Audio encoders transform raw waveforms or Fbank features into a high-level feature space that is informative for downstream tasks. These encoders can be classified into two categories: supervised models and self-supervised models. Supervised models typically employ ASR tasks to train an end-to-end model that includes an audio encoder and a text decoder, for instance, an attention-based decoder or a Connectionist Temporal Classification (CTC) decoder. By omitting the decoder, the encoder can serve as a feature extractor (Radford et al., 2022; Baevski et al., 2020). Self-supervised models can be trained on unlabeled speech signals. For instance, Wav2Vec2 (Baevski et al., 2020) and HuBERT (Hsu et al., 2021) were trained to predict the pseudo-discrete target at masked time steps. WavLM (Chen et al., 2022) is a variant of HuBERT, designed to facilitate speaker identity extraction by using multi-speaker signals. Different model architectures, training methods, and data can result in encoders with distinct properties and advantages, making the mixture of audio encoders effective for Speech LLMs.

Speech LLM: The native LLM handles only text modality. For the LLM to directly process audio modality, it must comprehend audio features, which contain significantly more information than transcriptions and are substantially longer than text embeddings. Since text is represented as discrete tokens, it is natural to extract discrete tokens from continuous speech signals and then expand the vocabulary of text LLMs to understand these speech tokens (Rubenstein et al., 2023; Veluri et al., 2024; Ma et al., 2024). An alternative is to use an adapter layer, to directly convert the continuous speech features to the continuous embedding space of the LLM. For example, QwenAudio (Chu et al., 2024) employs average pooling to downsample speech features, followed by two linear layers for projection. SALMONN (Tang et al., 2024) utilizes the Q-former (Yu et al., 2024), a cross-attention-based adapter, to achieve a higher compression ratio. Compared to previous works, our adapter handles more encoders and generates different features based on the prompt, rather than a single feature for all prompts.

Mixture of experts: With the introduction of the Transformer architecture (Vaswani et al., 2017), increasing model size has been shown to improve performance significantly (He et al., 2016; Kaplan et al., 2020). This has led to growing interest in the MoE approach, which replaces the FFN sub-layer in Transformer models with multiple experts, thereby enabling substantial parameter growth while maintaining constant inference costs (Shazeer et al., 2017). These MoE methods typically employ massive experts and extremely sparse activation routing, without explicitly considering the specialization of individual experts (Fedus et al., 2022; Lepikhin et al., 2020). However, the vast scale of these models presents significant deployment challenges. In contrast, the earliest MoE research introduced a data-dependent, trainable combining method (Jacobs et al., 1991; Masoudnia and Ebrahimpour, 2014), which aims to decompose complex tasks into simpler sub-tasks, each managed by a dedicated expert. Such works have inspired recent advances in developing modular models with emphasis on handling each task by a task-specific expert called expert specialization (Ma et al., 2018; Gupta et al., 2022), providing solutions for deploying large-scale MoE models (Lu et al., 2024) and enabling individual experts to learn and decompose diverse knowledge (Dai et al., 2024). Inspired by these insights, we proposed a specialized MoE fusion method integrating multiple audio features to enhance Speech LLMs.

7 Conclusion

In conclusion, we propose PaM, a MoE-based feature fusion method designed to provide Speech LLM with diverse features from multiple encoders based on users' input prompts. Experimental results indicate that PaM surpasses both single-encoder and multi-encoder baselines across a variety of tasks and datasets. We provide a detailed analysis of the feature importance of different encoders, demonstrating that PaM effectively leverages different encoders and levels of features for distinct tasks. Additionally, we present comprehensive experimental results for the selection and combination of encoders. For future work, we intend to expand the training data and incorporate additional tasks.

8 Limitations

Owing to resource constraints, our training data is limited to several hundred hours. It would be preferable to implement our method in larger-scale experiments to facilitate comparison with existing strong Speech LLMs such as Qwen-Audio (Chu et al., 2024) on a more comprehensive benchmark like AirBench (Yang et al., 2024).Additionally, we train the PaM module from scratch using a predefined list of audio encoders. It would be beneficial to investigate the addition of new encoders to an already trained Speech LLM to enhance its performance on new tasks or in new domains. We leave this for future work.

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. 2019. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proc. ACL*.

S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, et al. 2022. WavLM: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*.

Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, Wanxiang Che, Xiangzhan Yu, and Furu Wei. 2023. Beats: Audio pre-training with acoustic tokenizers. In *Proc. ICML*, pages 5178–5193.

Zhehuai Chen, He Huang, Andrei Andrusenko, Oleksii Hrinchuk, Krishna C. Puvvada, Jason Li, Subhankar Ghosh, Jagadeesh Balam, and Boris Ginsburg. 2024. Salm: Speech-augmented language model with in-context learning for speech recognition and translation. In *Proc. ICASSP*.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, Chang Zhou, and Jingren Zhou. 2024. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Yassir Fathullah, Chunyang Wu, Egor Lakomkin, Junteng Jia, Yuan Shangguan, Ke Li, Jinxi Guo, Wenhan Xiong, Jay Mahadeokar, Ozlem Kalinli, Christian Fuegen, and Mike Seltzer. 2023. Prompting large language models with speech recognition abilities. *arXiv preprint arXiv:2307.11795*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Shashank Gupta, Subhabrata Mukherjee, Krishan Subudhi, Eduardo Gonzalez, Damien Jose, Ahmed H Awadallah, and Jianfeng Gao. 2022. Sparsely activated mixture-of-experts are robust multi-task learners. *arXiv preprint arXiv:2204.07689*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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 Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *Proc. ICLR*.

Shujie Hu, Long Zhou, Shujie Liu, Sanyuan Chen, Lingwei Meng, 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. *arXiv preprint arXiv:2404.00656*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. 2019. Audiocaps: Generating captions for audios in the wild. In *NAACL-HLT*.

W. Kraaij, T. Hain, M. Lincoln, and W. Post. 2005. The AMI meeting corpus. *Proc. International Conference on Methods and Techniques in Behavioral Research*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Yuang Li, Jiawei Yu, Min Zhang, Mengxin Ren, Yanqing Zhao, Xiaofeng Zhao, Shimin Tao, Jinsong Su, and Hao Yang. 2024. Using large language model for end-to-end chinese asr and ner. In *Proc. Interspeech*.

Shaoshi Ling, Yuxuan Hu, Shuangbei Qian, Guoli Ye, Yao Qian, Yifan Gong, Ed Lin, and Michael Zeng. 2023. Adapting large language model with speech for fully formatted end-to-end speech recognition. *arXiv preprint arXiv:2307.08234*.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939.

Ziyang Ma, Yakun Song, Chenpeng Du, Jian Cong, Zhuo Chen, Yuping Wang, Yuxuan Wang, and Xie Chen. 2024. Language model can listen while speaking. *arXiv preprint arXiv:2408.02622*.

Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. LibriSpeech: an ASR corpus based on public domain audio books. *Proc. ICASSP*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*.

Paul K Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, et al. 2023. Audiopalm: A large language model that can speak and listen. *arXiv preprint arXiv:2306.12925*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun MA, and Chao Zhang. 2024. SALMONN: Towards generic hearing abilities for large language models. In *Proc. ICLR*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*, volume 30.

Bandhav Veluri, Benjamin Peloquin, Bokai Yu, Hongyu Gong, and Shyamnath Gollakota. 2024. Beyond turn-based interfaces: Synchronous llms as full-duplex dialogue agents. In *Proc. EMNLP*, pages 21390–21402.

Bin Wang, Xunlong Zou, Geyu Lin, Shuo Sun, Zhuohan Liu, Wenyu Zhang, Zhengyuan Liu, AiTi Aw, and Nancy F Chen. 2024. Audiobench: A universal benchmark for audio large language models. *arXiv preprint arXiv:2406.16020*.

Qian Yang, Jin Xu, Wenrui Liu, Yunfei Chu, Ziyue Jiang, Xiaohuan Zhou, Yichong Leng, Yuanjun Lv, Zhou Zhao, Chang Zhou, et al. 2024. Air-bench: Benchmarking large audio-language models via generative comprehension. *arXiv preprint arXiv:2402.07729*.

Wenyi Yu, Changli Tang, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. 2024. Connecting speech encoder and large language model for asr. In *Proc. ICASSP*.

Wenyu Zhang, Shuo Sun, Bin Wang, Xunlong Zou, Zhuohan Liu, Yingxu He, Geyu Lin, Nancy F. Chen, and Ai Ti Aw. 2024. Mowe-audio: Multitask audiollms with mixture of weak encoders. *arXiv preprint arXiv:2409.06635*.

A Appendix

A.1 Details of Models and Datasets

In this paper, we leverage multiple audio encoders and LLM to construct the end-to-end speech LLM. Our training dataset is sourced from commonly used open-source datasets, totalling approximately 450 hours of audio data, corresponding to 313,208 samples, as outlined in Table 5.

Table 5: The whole training datasets.

Data Source	Task	Hours	Sample
Librispeech-clean-100	ASR	100h	28539
AMI	ASR	100h	108502
Common Voice V4 (Part)	SNV	~150h	137041
Audio Caption	AC	100h	39126

In our paper, we adopt multiple pre-trained audio encoders and LLMs, we list the architecture setting for all model we used in our experiments in Table 6. Notably, for the Whisper model, we only used its encoder part as an audio feature extractor.

A.2 Details of Baseline Implement

In our work, we compare our method against two types of baselines. The first baseline consists of models using a single encoder, while the second baseline involves fusing multiple audio encoders either based on previous work (Hu et al., 2024; Tang et al., 2024) or through an averaging operation.

For the single encoder baseline, we train the model using the same settings as in our method. For the second baseline, we train the model using the open-source codebases from WavLLM and SALMONN, we integrate the adapter components from these repositories into our code and train the baseline model using our training data, employing the same pre-trained audio encoders and LLMs as in our method. During training, we applied the same hyperparameters as our method. Since we use different encoders and LLMs compared to the baselines, we adjust the dimensions of the adapter to match the specific audio encoder and LLM we used while maintaining other dimensions independent of the audio encoders and LLM unchanged. Notably, we trained the SALMONN with query length=32 (as training with the original setting query length=1 failed) to ensure comparable performance with the other baseline methods.

A.3 All result

The detailed results of our experiments with multiple encoders are summarized in Table 7. We observe that, in most cases, the audio encoder that performs well on a single task also enhances the performance of the fusion model on that task. In cases where performance degradation occurs on a specific task when using the corresponding encoder, the fusion model consistently includes the HuBERT audio encoder, suggesting that incorporating the HuBERT model may have a detrimental effect. This could be attributed to the fact that the HuBERT model is trained on a smaller pre-trained dataset compared to other audio encoders. Notably, even in this case, fusing four audio encoders yields comparable results to fusing three encoders on the AVG Rank, indicating that incorporating more encoders can still lead to performance improvements.

Audio Encoder Models	Enc Param	Layers	d_{model}	d_{ffn}	d_k	H	Norm
openai/whisper-small	88M	12	768	3072	64	12	Pre
microsoft/wavlm-base-plus	94M	12	768	3072	64	12	Post
facebook/wav2vec2-base-960h	94M	12	768	3072	64	12	Post
openai/whisper-medium	307M	24	1024	4096	64	16	Pre
microsoft/wavlm-large	315M	24	1024	4096	64	16	Post
facebook/wav2vec2-large-960h	315M	24	1024	4096	64	16	Post
Large Language Models	**Lora Param**	**Layers**	d_{model}	d_{ffn}	d_k	H	**Norm**
Qwen/Qwen2.5-3B	7M	36	2048	11008	128	16	Pre
Qwen/Qwen2.5-7B	10M	28	3584	18944	128	28	Pre
meta-llama/Llama-3.2-3B	9M	28	3072	128256	128	24	Pre
meta-llama/Llama-3.1-8B	13M	32	4096	14336	128	32	Pre

Table 6: The settings of pre-trained model we used in our experiments. For the audio encoder models, we utilize only the encoder component and freeze all parameters. For the LLMs, we freeze the base model parameters and apply LoRA adapters to fine-tune the model.

Encoders	Whisper	WavLM	Wav2Vec2	HuBERT	LibriSpeech		AMI	SNV	AudioCaps	AudioCaps QA	Avg		AVG Rank
					WER(clean)↓	WER(other)↓	WER↓	Acc↑	METEOR↑	METEOR↑	↓	↑	
1	√	-	-	-	9.61	16.73	16.27	18.8%	32.96	15.04	14.20	22.27	11.67
1	-	√	-	-	5.59	10.57	18.97	41.4%	27.14	12.77	11.71	27.10	11.50
1	-	-	√	-	4.30	09.46	26.69	39.0%	23.81	11.20	13.48	24.67	11.67
1	-	-	-	√	7.47	13.85	N	31.1%	23.94	11.28	N	22.10	14.00
Best-1					4.30	9.46	16.27	41.4%	32.96	15.04	13.13	24.04	
2	√	√	-	-	5.07	09.50	13.59	49.5%	**35.47**	16.16	9.38	33.71	06.00
2	√	-	√	-	3.82	07.55	13.51	38.4%	35.43	15.55	8.29	29.79	05.83
2	√	-	-	√	4.37	09.70	14.79	43.2%	35.33	16.62	9.62	31.72	06.50
2	-	√	√	-	3.17	06.76	19.60	**61.2%**	31.70	14.89	9.84	35.93	05.83
2	-	√	-	√	3.96	07.64	22.24	31.7%	30.28	13.99	11.28	25.32	10.33
2	-	-	√	√	3.27	07.29	21.43	35.8%	29.85	14.62	10.66	26.76	09.00
Best-2					3.17	06.76	13.51	61.2%	35.47	16.62	9.85	36.65	
3	√	√	√	-	3.65	07.07	**12.79**	42.8%	**35.47**	15.70	**7.83**	31.32	04.00
3	√	√	-	√	4.42	10.26	13.47	47.4%	35.32	16.62	9.38	33.11	06.50
3	√	-	√	√	4.58	06.99	15.11	59.1%	35.33	16.62	8.89	**37.02**	05.50
3	-	√	√	√	**3.09**	**06.64**	22.80	26.0%	31.90	15.14	10.84	24.35	07.67
Best-3					3.09	06.64	12.79	59.1%	35.47	16.62	9.24	31.45	
4	√	√	√	√	3.94	07.28	14.06	57.3%	35.37	**16.79**	8.43	36.49	04.00

Table 7: Detailed results of incorporating different combinations of audio encoders.