TIME COMPRESSION FOR AUDIO MOMENT RETRIEVAL WITH LARGE AUDIO LANGUAGE MODELS

Technical Report

Hiroshi Nishijima[1]*, Daisuke Saito*[1]*, Nobuaki Minematsu*[1]

[1]The University of Tokyo, Japan.
{hiroshi, dsk_saito, mine}@gavo.t.u-tokyo.ac.jp

ABSTRACT

We study time compression (fast-forwarding) as a test-time adaptation that brings long-form audio into the short native context of a large audio language model (LALM), for the DCASE 2026 Task 6 problem of Audio Moment Retrieval (AMR) from long audio. Instead of redesigning the model for long inputs, we time-stretch each recording so that the entire clip fits into a single native input window. We then adapt the model with staged fine-tuning: it first learns temporal grounding, and a final stage fine-tunes it on the target data. Our best system is a three-stage fine-tuned Qwen2-Audio-7B. At inference, we compress each recording to at most 15 s, half of the model's 30 s input limit, and rerank its predictions across several compression settings. On CASTELLA it reaches R1@0.7 (Recall@1 at IoU 0.7) of 27.01 on validation and 20.91 on test, and it exceeds a DETR-based baseline model that ingests the whole recording without any compression. We further show that pitch-preserving compression is essential and that compressing the whole clip beats sliding-window inference for short-context LALMs. The best compression amount tends to track the length of the model's training clips, a trend that is significant on the validation split.

***Index Terms*—** Audio moment retrieval, Large Audio Language Models, Time Compression, DCASE 2026 Task 6

1. INTRODUCTION

Audio Moment Retrieval (AMR) is the task of finding the time interval in a recording that matches a natural-language query [1]. The DCASE 2026 Task 6 setting is demanding because the recordings are long: in its benchmark, CASTELLA [2], they run up to about five minutes, while the moment being searched for is short. Such durations far exceed the native context of current large audio language models (LALMs), whose Whisper-style encoders take in only about 30 s at a time [3, 4]. To cover a longer recording the model must therefore read it as a sequence of 30 s windows, and this windowing breaks the recording into pieces and discards the global context, which makes it hard to retrieve a short moment that may lie anywhere in a long clip.

We study a deliberately minimal alternative to windowing, architectural redesign, and retraining for long contexts: time compression. We fast-forward the audio so that the entire recording fits into a single 30 s window before it reaches the model, which keeps the global context that windowing would lose, at the cost of temporal resolution. This turns long audio retrieval into a test-time adaptation that remaps a long input onto the timescale the encoder learned to handle. We describe a staged fine-tuning recipe that adapts the model to the compressed inputs, and we report an ablation of which ingredients help and which do not. Our best submitted system is a three-stage Qwen2-Audio-7B that we evaluate with the clip compressed to 15 s and rerank across several compression settings. It reaches R1@0.7 of 27.01 on validation and 20.91 on test, the best CASTELLA scores among the LALMs we tried, and it exceeds the QD-DETR-based baseline reference.

2. METHOD

2.1. Time compression

Let an input recording have duration d seconds, and let m be the target length to which we compress it. We compress by an integer factor

$$k = \lceil d/m \rceil , \tag{1}$$

so that the compressed clip is at most m seconds long and fits into a single forward pass. The factor is adaptive: short clips are barely altered, whereas longer clips are compressed more aggressively. By default, m equals the model's native context of about 30 s; smaller targets compress the clip below the native context. We identify a configuration by its target length, for example 30 s or 15 s, and for some control experiments we instead fix the factor k directly.

We compare two operators that realize the speedup in different ways. Stretching applies a phase-vocoder time-stretch that accelerates playback by k but preserves pitch, so that spectral content is shifted in time but not in frequency. Decimation resamples the waveform and plays it back at the original sampling rate, which accelerates playback by k but also raises pitch by the same factor. The two operators are acoustically very different, and this distinction is important in practice: some cues survive stretching far better than they survive decimation.

Compression also requires a consistent convention for time coordinates. During training, we scale each ground-truth boundary by $1/k$ so that the supervision matches the compressed waveform, and at inference we multiply every predicted boundary by k so that it returns to the original timeline.

The intersection-over-union (IoU) between two intervals is invariant to a common multiplicative rescaling of the time axis, so this remapping leaves all IoU-based metrics unchanged, and the only effect of compression on the score is what the model perceives and predicts.

Finally, for inputs longer than m a model typically adopts one of two strategies (Fig. 1). The first strategy compresses the entire clip into a single m-second input, which preserves global context at reduced temporal resolution. The second slides an m-second window along the recording and stitches the per-window predictions, which keeps the resolution but discards context across windows and

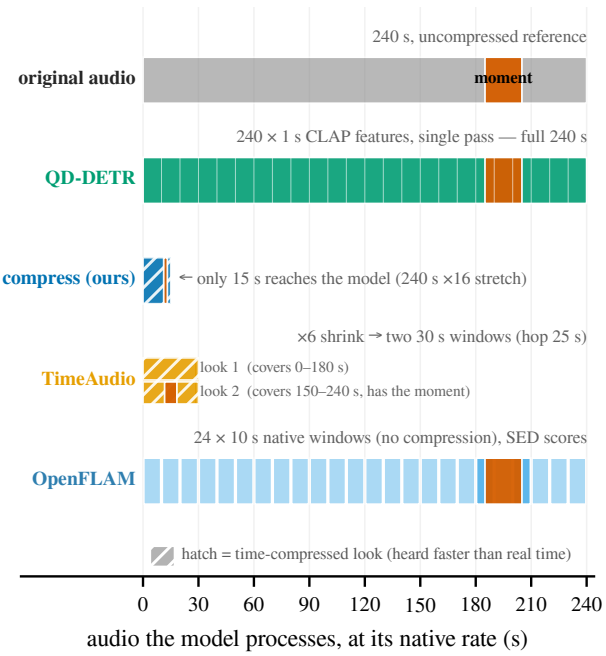

Figure 1: How a 240 s recording reaches each model. Our model compresses the whole clip into one 30 s input; windowing stitches eight 30 s windows.

multiplies the inference cost. Our models compress the whole clip unless we state otherwise, and we include windowing as an explicit baseline so that we can isolate the effect of compression.

2.2. Query prompting and interval decoding

Prompting. We insert each query into a fixed question template:

> *Between which timestamps does the audio related to '<query>' occur?*

The question and the compressed audio are wrapped in the model's standard chat template as a single user turn. We use the same question and answer format when we fine-tune the model, and the supervision targets follow its answer convention ("*The given query happens in s to e.*", and further pairs are listed when several moments match), so the input format is identical between fine-tuning and inference. We use the same wording for Qwen2-Audio and for SALMONN, and only the surrounding chat template differs between the two models.

Interval decoding. We decode greedily, with no sampling and at most 96 new tokens, and we extract every "s to e" pair from the generated text with a regular expression. We discard any pair with $e \leq s$. The model usually emits between one and seven intervals per query. By default we keep the generation order as the ranking: the i-th interval receives a pseudo-score of $1 - 0.01\,i$, so the first interval that the model writes is ranked highest. R1@θ then depends only on the top-ranked interval, whereas mAP consumes the full ranked list. Section 2.4 describes an optional reranking step that reorders the candidates, and we use it in some of our submitted systems. Because the model rarely emits more than a few intervals, its mAP is structurally lower than that of a frame-based detector that scores every segment. We map all boundaries back to the original timeline by the compression factor k of Eq. (1). If no pair can be parsed, we fall back to a single prediction that spans the whole clip with zero score.

2.3. Staged adaptation

We apply up to three fine-tuning stages on top of a pre-trained model, and each stage closes a distinct gap between the base model and the target task.

Stage 1 (grounding) fine-tunes the model on a large corpus of short clips, about 10 s each, where each clip is labeled with the time interval that matches a query (Section 3.1). The clips already fit the native context, so this stage applies no compression. It teaches the form of the task, and it does not yet use the long recordings of the target benchmark.

Stage 2 (compression augmentation) continues fine-tuning and fast-forwards each training example by a factor drawn at random from $\{1, 2, 4, 6, 8\}$, which shows the model compressed audio during training and makes it robust to the time-warping it will meet at evaluation.

Stage 3 (in-domain) fine-tunes on the training split of the target benchmark, with each recording compressed to the native 30 s target. At evaluation we compress more aggressively, to a 15 s target, so the model is trained at a moderate compression and tested under a stronger one; Section 4.3 shows that this asymmetry beats matching the two.

We additionally study three variants. A reinforcement-learning branch (Group Relative Policy Optimization; GRPO) [5] fine-tunes a model further with a reward built from temporal overlap. A second variant skips Stage 2 and runs Stage 1 then Stage 3, which tests whether the compression augmentation stage adds anything. A third trains on the target benchmark alone from the base model, with no grounding.

2.4. Reranking across compression settings

The amount of compression is itself a cheap source of test-time augmentation. We run the same model at several target lengths, which gives several candidate interval sets for one query, and we rerank the pooled candidates by how well they agree across these runs. An interval that recurs at a similar place under different compressions is more trustworthy than one that appears only once. The only extra cost is one inference pass per setting, with no further training and no second model. This reranking is one of our challenge submissions, alongside the single-pass system and two systems that add extra training data. Section 4.1 reports the gain that the reranking provides over a single pass.

3. EXPERIMENTAL SETUP

3.1. Data

CASTELLA [2] is a dataset for Audio Moment Retrieval: it pairs long real-world recordings with natural-language queries and human-annotated temporal boundaries for the moments they describe. Its recordings span roughly 60 to 300 s, and the target moments have a median width of only a few seconds and concentrate toward the end of each clip, which makes both reading at coarse resolution and reading by truncation particularly fragile. Stage 3 uses the 871 training clips, which expand to 2,161 query and moment supervision items. **FTAR** [6] is a temporal audio-grounding corpus

introduced with TimeAudio. For Stage 1 we use 109,533 query and moment pairs over 61,576 clips of mean duration about 10 s; the data-expansion variant additionally inverts its dense audio captions into grounding pairs.

3.2. Models and training conditions

Our primary LALM is **Qwen2-Audio-7B**, and we use **SALMONN-7B** as a cross-model check under an identical compression protocol. As references, we report three systems that are not based on a general-purpose LALM: **QD-DETR** [7, 1], a detection-transformer (DETR) detector that consumes full-length 1 s/1 s CLAP features [8, 9] without any compression; **OpenFLAM/FLAM** [10], a frame-wise open-vocabulary sound-event-detection (SED) model evaluated with 10 s windows; and **TimeAudio** [6], a time-token LALM that operates in a hybrid compress-and-window regime. We implement the compression operators with a phase-vocoder time-stretch (stretch) and with resampling (decimate), and the default target length is m=30 s, which matches the Whisper-style front ends.

All LALM fine-tuning uses low-rank adaptation (LoRA) [11] with AdamW (no weight decay), a linear warmup and then a cosine decay to zero, gradient clipping at norm 1.0, and bf16 precision.

For Qwen2-Audio the adapters have rank 16 and α=32 over every attention and feed-forward projection (44M trainable parameters). Stage 1 runs for one epoch at learning rate 1×10^{-4}, Stage 2 for half an epoch at 5×10^{-5}, and Stage 3 for three epochs at 1×10^{-5} (540 optimizer steps on the CASTELLA set), with an effective batch size of 16 for the first two stages and 12 for the third. The data-expansion variant keeps these settings and changes only the Stage 1 corpus.

SALMONN uses a separate recipe: LoRA rank 8 over the same projections, a backbone learning rate of 1×10^{-5} with a smaller 2×10^{-6} rate for the audio connector, weight decay 0.05, and a warm start from the TimeAudio stack so that its time-stamping behavior is initialized rather than learned from scratch. It follows the same one, half, and three epoch schedule across the three stages at an effective batch size of 12.

The GRPO branch starts from the matching fine-tuned adapter and optimizes a reward of the form $\mathrm{IoU} + 0.1\, f_{\mathrm{fmt}} + 0.3\, f_{\mathrm{overlap}}$, where f_{fmt} rewards well-formed time stamps and f_{overlap} rewards coarse temporal overlap. It draws 8 samples per query at learning rate 1×10^{-6} with a KL penalty of 0.04 to the frozen start and an effective batch of 4, and it runs for 2,500 steps from the two grounding starts and 1,080 from the in-domain start.

We train on NVIDIA GH200 GPUs with one GPU per fine-tuning stage, so the three-stage pipeline takes about 12.5 GPU-hours, whereas GRPO runs data-parallel on four GPUs. Our submitted systems use Qwen2-Audio as the only pretrained backbone, and FTAR is their only external training data. The cross-model and reference systems additionally build on SALMONN, TimeAudio, FLAM, and CLAP.

3.3. Evaluation protocol

We report Recall@1 at an IoU threshold of 0.7 (R1@0.7) together with mean average precision (mAP@avg), and we compute both with the official Task 6 scorer on the CASTELLA validation and test splits. R1 measures whether the single top-ranked prediction hits the target at the given IoU, whereas mAP rewards a well-ranked set of candidate intervals, and the two are complementary. For every comparison that we label significant, neutral, or equivalent, we use a query-level paired bootstrap on R1@0.7 with 10,000 resamples at a fixed seed. We treat a difference as significant when its 95% confidence interval excludes zero. At inference each LALM compresses the clip with the adaptive factor of Eq. (1) and decodes intervals as described in Section 2.2.

Table 1: Main comparison on CASTELLA (%): the best configuration of each model family. Shaded: our primary submission. TimeAudio is shown after continued fine-tuning at k=6.

	Val		Test	
System	R1@0.7	mAP@avg	R1@0.7	mAP@avg
QD-DETR	20.74	16.78	14.70	11.81
OpenFLAM	10.63	8.03	11.38	8.39
FLAM (SED-FT)	10.92	9.38	11.98	9.65
TimeAudio	12.64	6.83	7.37	4.98
SALMONN				
(grounding only)	10.63	6.86	7.74	5.13
(2-stage)	14.94	9.98	12.95	8.70
Our submitted Qwen2-Audio				
(3-stage)	24.14	16.99	18.68	13.33
+rerank	**27.01**	**18.55**	**20.91**	**13.87**

4. RESULTS

4.1. Main comparison

Table 1 reports the best configuration of each model family. Our submitted Qwen three-stage system attains the strongest LALM scores on both splits and beats the long-form QD-DETR reference on R1@0.7 on both validation and test, and it does so even at a single pass, before any reranking. This gain comes from compression and adaptation alone, because the model never sees more than a single 30 s input window, not from any change in architecture or context length. The same recipe transfers to a second LALM, SALMONN, which improves by 5.21 test points over the grounding stage alone (test 7.74 to 12.95, both in Table 1). We submit four systems, listed in Table 2. Two use the grounding corpus only: a single forward pass, and a cross-setting vote that reruns this same model at five target lengths and keeps the intervals that agree across them, which adds neither training nor a second model. The other two add the expanded grounding set. For the few queries whose predicted start falls beyond the clip, they substitute a no-ext prediction. We fix every submission choice on validation. The cross-setting vote is our primary submission (27.01/20.91); the two data-expansion systems reach a higher test R1@0.7 (up to 21.88) but lose on validation.

4.2. Which training stages matter

Table 3 isolates the contribution of each Qwen stage at a fixed 30 s setting, so that any difference reflects training rather than compression at evaluation time. Two ingredients dominate. The grounding stage (Stage 1) is required: the model that trains on CASTELLA alone fails to acquire the task at all and scores only 1.15 R1@0.7 on validation, which is below the 4.02 of the untuned model, because

Table 2: Our four submitted systems (CASTELLA, %). The data-expansion systems (S3, S4) repair overshooting predictions with a no-ext one, from S1 for S3 and from S2 for S4.

	Val		Test	
Submitted system	R1@0.7	mAP@avg	R1@0.7	mAP@avg
Grounding only (no-ext)				
S1: vote rerank	27.01	18.55	20.91	13.87
S2: single pass	24.14	16.99	18.68	13.33
Data expansion (data-ext) + overshoot repair				
S3: repair (vote)	24.43	16.31	21.88	14.81
S4: repair (single)	24.14	16.37	21.43	14.66

Table 3: Qwen2-Audio training-stage ablation (CASTELLA, %). The 30 s setting unless a 15 s eval is noted; cumulative rows add the listed stage to the row above.

	Val		Test	
Training	R1@0.7	mAP@avg	R1@0.7	mAP@avg
Cumulative stages				
Zero-shot	4.02	3.60	2.01	1.66
+ Stage 1	9.77	5.09	6.77	4.25
+ Stage 2	9.77	4.65	7.96	5.33
+ Stage 3	18.97	13.52	16.74	12.28
Training-order variants				
CASTELLA only	1.15	0.95	1.04	0.81
Grounding CASTELLA	19.54	13.08	16.96	11.70
evaluated at 15 s	18.97	14.73	17.41	12.54
GRPO and data expansion				
GRPO				
on Stage 1	9.48	4.78	6.32	4.00
on Stage 2	10.63	4.94	7.74	5.02
on three-stage	19.54	13.09	16.37	11.76
Data expansion				
(Stage 1)	9.77	4.89	8.48	5.09
(full pipeline)	22.41	14.67	20.31	14.10

the 2,161 in-domain items are far too few to teach time stamping from scratch. Stage 1 on the large grounding corpus instead lifts validation R1@0.7 from 4.02 to 9.77. The in-domain fine-tuning (Stage 3) is then the largest single contributor: it roughly doubles R1@0.7 on both splits (validation 9.77 to 18.97, test 7.96 to 16.74). Compression augmentation (Stage 2) is neutral at this 30 s setting: the direct variant that skips it even slightly exceeds the full three-stage model on R1@0.7. Its value appears only at the more aggressive submission setting. There the full three-stage model adds about 1.3 test points over the skip-Stage 2 variant (18.68 versus its 17.41 at the 15 s eval row of Table 3). This gap is consistent in direction but not significant, so we read Stage 2 as a source of robustness to overcompression rather than of accuracy at the native scale. Reinforcement learning is neutral from every starting point: GRPO leaves the grounding model, the model with compression augmentation, and the full three-stage model all statistically unchanged. Data expansion does not help on its own either: it adds only about a point at Stage 1. Through the full pipeline, however, it matches the pipeline without extra data and is statistically equivalent at the submission setting, so data expansion helps only in combination with the in-domain fine-tuning and never in isolation.

4.3. Which compression choices matter

Figure 2 sweeps the target length, and Table 4 isolates the other choices. The sweep traces a U-shape in R1@0.7: moderate overcompression helps, and each model peaks near the duration of its own training clips, the grounding model near 10 s and the three-stage model near 15 s. The best target length therefore tracks the duration of the training clips. This is consistent with the idea that compression remaps the input onto the model's trained time range, although we cannot rule out a simpler adaptation to the compressed inputs. This U-shape is significant on validation but not on test, so we report the overcompression benefit as a trend that holds on validation.

The operator and the strategy for long inputs matter just as much. Pitch-preserving stretching is far better than decimation (Table 4), which suggests the model relies on cues that carry pitch and resemble speech; decimation destroys these. Sliding-window inference drops sharply (validation R1@0.7 of 1.15), so a short-context model should compress the whole clip rather than window it; by contrast, a model without a native limit, such as the long-form QD-DETR, only degrades under compression. Finally, in our controlled comparison, an asymmetric setting that trains at 30 s and evaluates at 15 s beats matching the two target lengths, which shows the benefit of moderate compression at training with stronger compression at evaluation.

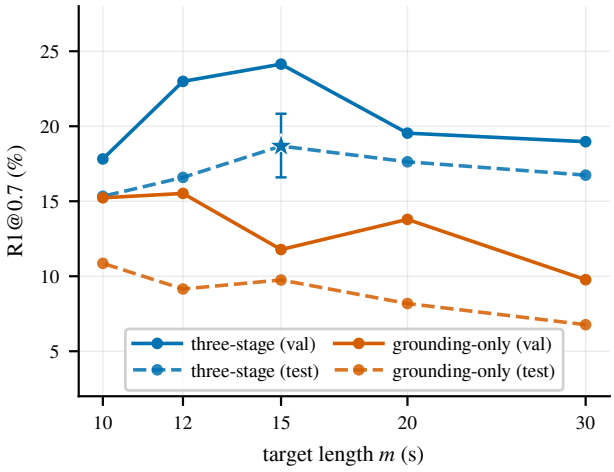

Figure 2: R1@0.7 versus the target length m. Each model forms a U-shape whose peak shifts to shorter m for models trained on shorter clips (the range-fit effect). The star marks the submitted operating point (m=15 s).

Table 4: Compression controls beyond the sweep of Fig. 2 (CASTELLA, %): stretch versus decimation, and matching training to evaluation compression.

	Val		Test	
Compression	R1@0.7	mAP@avg	R1@0.7	mAP@avg
Three-stage model (trained on 15 *to* 30 *s clips)*				
30 s setting				
stretch	18.97	13.52	16.74	12.28
decimation	14.66	10.53	9.08	6.56
train + eval at 15 s	19.83	13.30	17.78	12.62
Grounding model (trained on ≈10 *s clips)*				
30 s setting				
stretch	9.77	5.09	6.77	4.25
decimation	8.62	4.62	4.61	3.11

5. ANALYSIS

Two parts of the recipe account for most of the gain: the grounding stage and the in-domain fine-tuning. Reinforcement learning (GRPO), data expansion, and compression augmentation each add nothing on their own. How the audio is compressed matters too: pitch-preserving stretching beats decimation, and compressing the whole clip beats windowing. The best target length tends to track the duration of the training clips, and training at a moderate 30 s while evaluating at 15 s beats matching the two. The recipe also transfers to a second model, SALMONN. Because our models emit only a few candidate intervals, their mAP is low relative to R1; a light top-K post-processing recovers some mAP without changing R1. We also tried reranking by audio-text (CLAP) similarity, but it does not improve the top prediction, so we did not submit it.

6. CONCLUSION

We presented a time-compression recipe that lets a short-context audio language model handle long-form Audio Moment Retrieval. Our submitted Qwen2-Audio system compresses each recording to 15 s and reranks across compression settings, reaching R1@0.7 of 27.01 on validation and 20.91 on test on CASTELLA and surpassing a long-form DETR reference on test. The design choices that matter are grounding plus in-domain fine-tuning, pitch-preserving compression of the whole clip, and a target length close to that of the training clips on validation, with more aggressive compression at evaluation than at training.

7. ACKNOWLEDGMENT

This research is partially supported by Initiative on Recommendation Program for Young Researchers and Woman Researchers, Information Technology Center, The University of Tokyo.

8. REFERENCES

[1] H. Munakata, T. Nishimura, S. Nakada, and T. Komatsu, "Language-based audio moment retrieval," in *IEEE International Conference on Acoustic, Speech, and Signal Processing*, 2025.

[2] H. Munakata, I. Takehiro, T. Nishimura, and T. Komatsu, "Castella: Long audio dataset with captions and temporal boundaries," *arXiv preprint arXiv:2511.15131*, 2025.

[3] Y. Chu, J. Xu, Q. Yang, H. Wei, X. Wei, Z. Guo, Y. Leng, Y. Lv, J. He, J. Lin, *et al.*, "Qwen2-audio technical report," *arXiv preprint arXiv:2407.10759*, 2024.

[4] C. Tang, W. Yu, G. Sun, X. Chen, T. Tan, W. Li, L. Lu, Z. Ma, and C. Zhang, "Salmonn: Towards generic hearing abilities for large language models," in *International Conference on Learning Representations*, vol. 2024, 2024, pp. 16 607–16 629.

[5] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[6] H. Wang, Y. Li, S. Ma, H. Liu, and X. Wang, "Timeaudio: Bridging temporal gaps in large audio-language models," *arXiv preprint arXiv:2511.11039*, 2025. [Online]. Available: https://arxiv.org/abs/2511.11039

[7] W. Moon, S. Hyun, S. Park, D. Park, and J.-P. Heo, "Query-dependent video representation for moment retrieval and highlight detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 23 023–23 033.

[8] B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, "Clap learning audio concepts from natural language supervision," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[9] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[10] Y. Wu, C. Tsirigotis, K. Chen, C.-Z. A. Huang, A. Courville, O. Nieto, P. Seetharaman, and J. Salamon, "FLAM: Frame-wise language-audio modeling," in *Forty-second International Conference on Machine Learning (ICML)*, 2025. [Online]. Available: https://openreview.net/forum?id=7fQohcFrxG

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.