오디오 언어모델의 경량 모델링 레서피 탐구 (ASR&AAC)

김보현[1] 김재환[1] 김진석[1] 박진영[1] 성기훈[1] 양호철[1]
[1] Nota 02 team, Naver boostcamp ai tech CV13
{bhkim4550, mevest71, 99wlstjr, singiri12900, staragnarok, hocheol0303}@gmail.com

Abstract

오디오 언어 모델의 경량화는 메모리 및 연산 비용 절감과 실시간 응용을 가능하게 하는 중요한 연구 주제이다. ASR(Automatic Speech Recognition) 및 AAC(Audio-assisted Captioning) 작업을 위한 최적화된 경량 모델링 기법을 탐구하였다.

기존 접근법의 한계점을 분석하고, LLM(Large Language Model) 기반 음성 및 오디오 처리 모델을 다양한 경량화 기법(양자화, 지식 증류, LoRA 등)과 결합하여 최적화하는 방안을 실험적으로 검토하였다. 또한 Whisper 및 BEATs 기반의 Speech/Audio Encoder를 활용하여 다양한 데이터셋(GigaSpeech, LibriSpeech, Clotho 등)에서의 성능을 평가하였다.

실험 결과, 특정 경량화 기법이 모델 성능에 미치는 영향과 하드웨어 환경과의 상호작용이 성능 저하를 유발할 수 있음을 확인하였다.

이 프로젝트 결과는 제한된 연산 자원을 고려한 오디오 언어 모델 최적화 전략을 제시하며, 향후 실시간 오디오 처리 모델 개발을 위한 기초 연구로 활용될 수 있다.

소스 코드는 다음 GitHub 링크에서 확인할 수 있다.https://github.com/boostcampaitech7/level4-cv-finalproject-hackathon-cv-13-lv3

1 Introduction

최근 음성 어댑터의 결합과 사전학습(pretraining) 기법의 발전으로, 언어 모델은 음성, 음악, 환경음 등 다양한 오디오 신호를 효과적으로 이해하고 처리할 수 있게 되었다. 이러한 기술 발전은 자동 음성 인식(ASR), 오디오 캡셔닝(AAC) 등 여러 오디오 이해 작업에서 새로운 가능성을 열어주고 있다. 하지만 최신 오디오 모델들은 일반적으로 대용량 VRAM과 많은 계산 자원을 필요로 하여, 하드웨어 제약이 있는 기기에서는 적용이 어려운 문제가 있다.

특히 리소스가 제한된 환경에서는 경량화되면서도 높은 정확도를 유지하는 모델이 요구된다. 현재 널리 사용되고 있는 Salmonn과 같은 기존 접근 방식은 다단계 처리와 사전학습 전략을 통해 우수한 성능을 보였으나, 3단계 구조는 ASR이나 AAC와 같이 주로 2단계 처리가 필요한 작업에 대해 불필요한 복잡성을 초래할 수 있습니다. 또한, Whisper와 같이 단독으로 음성 인식을 수행하는 모델들도 존재하지만, 이들 모델은 전체 아키텍처가 무겁고 추론 지연(latency) 및 메모리 사용 측면에서 최적화가 필요한 상황이다.

이에 본 프로젝트에서는 음성 어댑터의 결합 및 사전학습 기법을 활용하여, 제한된 VRAM 환경에서도 효율적으로 동작할 수 있는 경량 오디오 언어 모델을 제안한다. 제안하는 모델은 다음과 같은 목표를 가지고 설계되었다.

1. 컴팩트한 오디오 인코더와 특화된 언어 모델을 결합하여 다양한 오디오 입력을 효율적으로 처리한다.
2. LoRA, 양자화(quantization), 지식 증류(Knowledge Distillation), 프루닝(pruning) 등의 최적화 기법을 도입하여 모델 크기와 추론 지연을 줄이는 동시에, ASR 및 AAC와 같은 주요 오디오 이해 벤치마크에서 경쟁력 있는 정확도를 유지한다.
3. 제한된 VRAM 환경에서도 우수한 성능을 발휘할 수 있도록 모델의 경량화를 달성한다.

2 Tools

2.1 Audiomentations

오디오 데이터 증강을 위해 사용할 수 있는 라이브러리다. 다양한 오디오 증강 기법(잡음 추가, 피치 변환, 속도 변환 등)을 간단한 함수 호출로 적용할 수 있어, 개발 및 실험 효율을 높일 수 있다.

3 프로젝트 팀 구성 및 역할

- **최적화:** 김진석, 성기훈
- **모델:** 김재환, 박진영
- **데이터:** 김보현, 양호철

4 프로젝트 수행 절차 및 방법

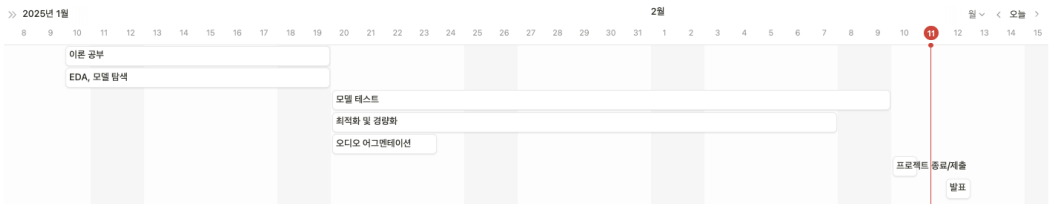

Figure 1. 프로젝트 진행 절차 개략도

프로젝트는 크게 (1) 데이터 분석 및 전처리, (2) 모델 선정 및 구성, (3) 최적화/경량화, (4) 결과 평가의 단계로 진행되었다.

5 EDA & Data Preprocessing

오디오 데이터에 대한 처리가 생소했기에 데이터셋 특징, 오디오 신호의 특성, 증강 기법 등을 우선적으로 학습했다. 이후 EDA 및 전처리를 통해 노이즈나 무음 구간을 정제하고, 학습에 적합한 형태로 만들었다.

5.1 데이터셋 특징

5.1.1 GigaSpeech

GigaSpeech는 10,000+ 시간의 대규모 영어 스피치 데이터다. 팟캐스트, 강연, 대화 등 다양한 도메인을 포함하며, 발화 단위 텍스트 전사가 존재한다. 다양한 억양, 소음, 발화 속도의 편차가 큰 편이다.

5.1.2 LibriSpeech

LibriSpeech는 1,000시간 분량의 정제된 낭독 음성 데이터로, 비교적 표준 발음이 일관되어 있고 WER(Word Error Rate)가 낮아 ASR 및 인코더 학습에 적합하다.

5.1.3 Clotho

Clotho는 15 30초 길이의 환경음, 음악, 혼합소리를 포함하는 캡셔닝 벤치마크로, 각 오디오당 사람이 작성한 5개 캡션을 제공한다.

5.1.4 MusicNet

MusicNet은 클래식 음악 녹음과 악기, 음표 단위 라벨이 있는 데이터셋이다. 다양한 악기에 대해 정확한 음표-시간 정렬 정보를 제공한다.

5.1.5 WavCaps

WavCaps는 다양한 도메인의 오디오 캡셔닝 데이터셋으로, 자동 생성된 캡션을 포함한다. 파일 길이와 캡션 길이에 대한 분포가 다양하다.

5.1.6 AudioCaps

AudioCaps는 YouTube에서 수집된 오디오와 그에 대응하는 캡션으로 구성된 데이터셋이다. 배경음 종류가 다양하다.

5.2 오디오 데이터의 특징

Waveform

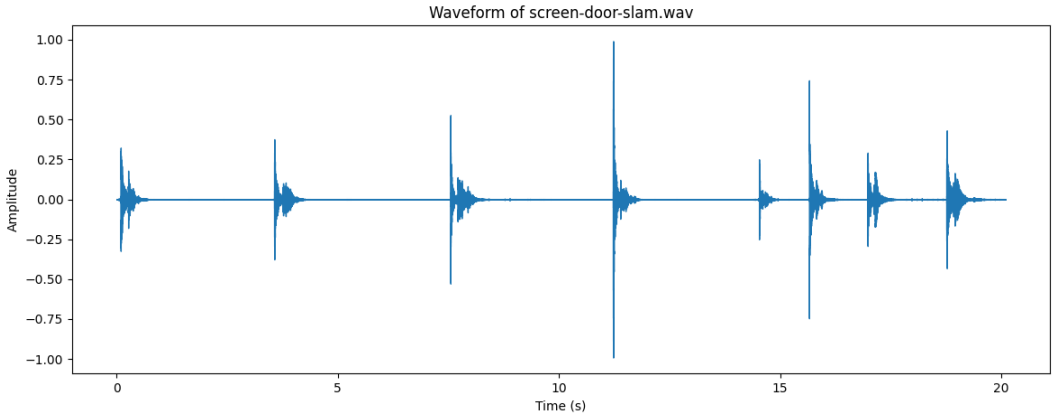

Figure 2. waveform of screen-door-slam.wav

시간에 따른 진폭 변화를 그래프로 표현하는 방법. 무음 구간, 클리핑, 노이즈 등을 시각적으로 파악할 수 있다.

Spectrogram

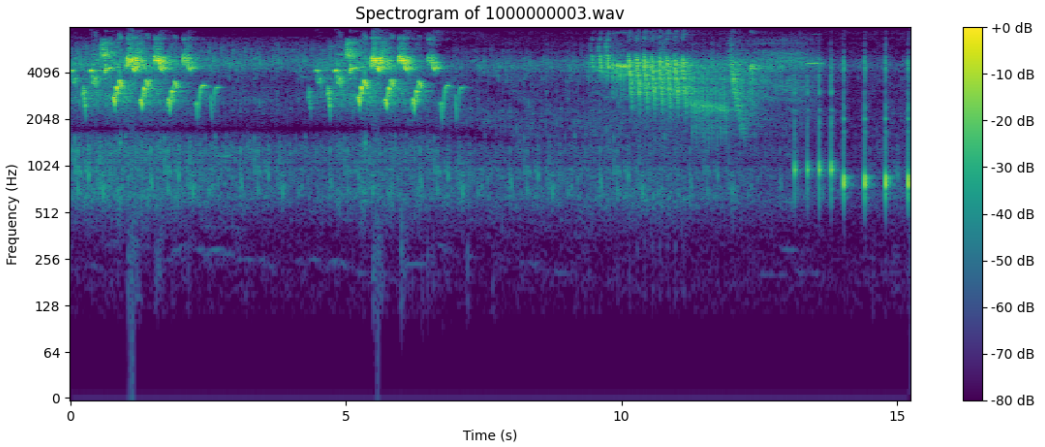

Figure 3. spectogram of 1000000003.wav

푸리에 변환으로 시간-주파수 변화를 나타내는 2차원 그래프. 특정 주파수 대역의 에너지가 높은 부분을 식별해 음성 분석이나 음향 분석에 활용한다.

RMS Energy

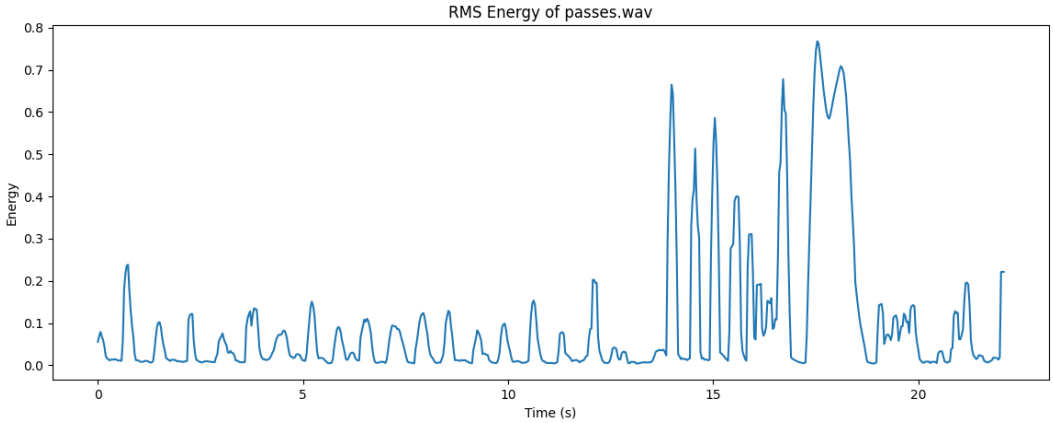

Figure 4. RMS Energy of passes.wav

신호의 에너지를 나타내는 값으로, 볼륨(음량)의 크기를 표현한다. 감정 분석, 특정 구간의 음성 강도 파악 등에 유용하다.

Zero-Crossing Rate (ZCR)

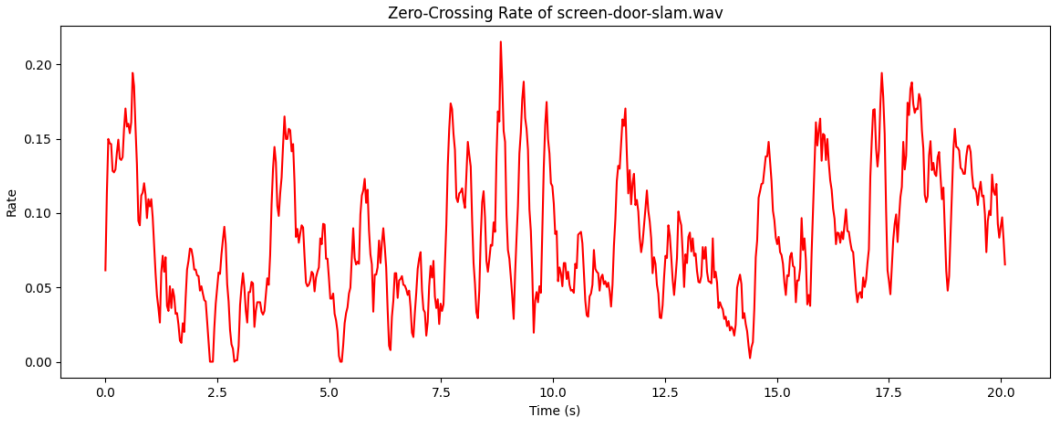

Figure 5. Zero-Crossing Rate of screen-door-slam.wav

신호가 0을 기준으로 부호가 바뀌는 횟수를 표현한다. 고주파 성분이나 잡음 검출에 활용될 수 있다.

Frequency Distribution

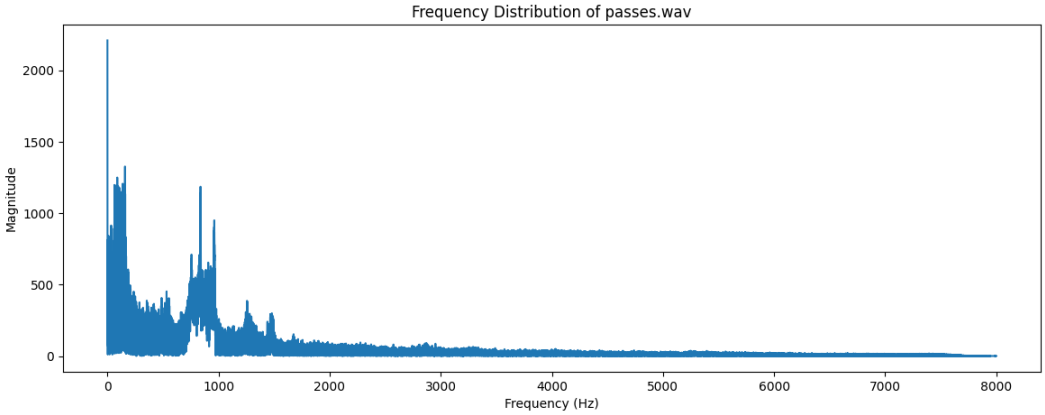

Figure 6. Frequency Distribution of passes.wav

주파수 대역별 에너지를 파악하여, 특정 인물의 음역대나 배경 소음의 주파수 대역 등을 확인할 수 있다.

5.3 데이터 분석

- Stage 1에는 약 150만 개의 annotation이, Stage 2에는 약 120만 개의 annotation이 있었다. - 여러 데이터셋에서 공통적으로 오디오 파일 하나가 ASR, AAC 등의 서로 다른 task로 활용되는 경우가 있었고, 노이즈 정도나 품질 편차도 컸다. - GigaSpeech에는 상대적으로 노이즈가 많고, LibriSpeech는 음질이 깨끗하다는 점을 확인했다.

5.4 데이터 전처리

- 0초짜리 오디오, 무음 데이터 등을 제거했다. - 멀티 채널 오디오의 경우 1채널로 합쳤다. - VAD(Voice Activity Detection)로 비음성 구간을 Trim - CLAP을 활용해 텍스트-오디오 유사도가 낮은 annotation을 제거(WavCaps, Clotho 등) - GigaSpeech는 데이터가 매우 많아, 유튜브/팟캐

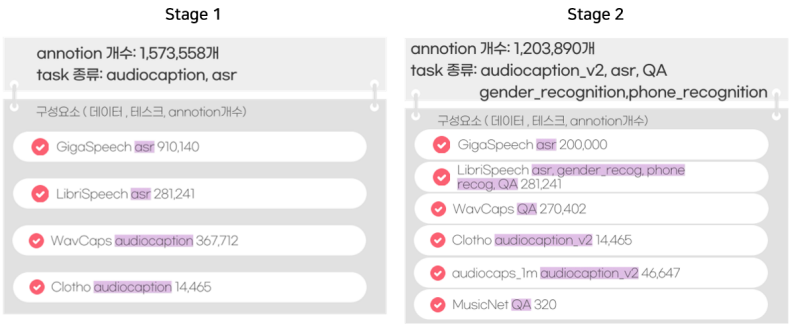

Figure 7. 데이터셋 분석

스트 도메인을 우선적으로 제거 후 랜덤 필터링으로 축소했다. - 최종적으로 stage1 약 77만 개, stage2 약 66만 개 annotation 남겼다

5.5 증강

Noise, Normalize, Trim, Time Stretch, Pitch Shift, Time Mask, IR(Impulse Response) 등 여러 증강 기법을 검토했으나, 실험 시간이 부족해 대부분이 미세한 성능 개선에 그쳤다. 결국 Normalize만 사용하기로 결정했고, ASR vs 비음성(AAC) 태스크별로 세밀한 증강 적용 전략을 시도했지만, 큰 성능 향상은 없었다.

6 Model

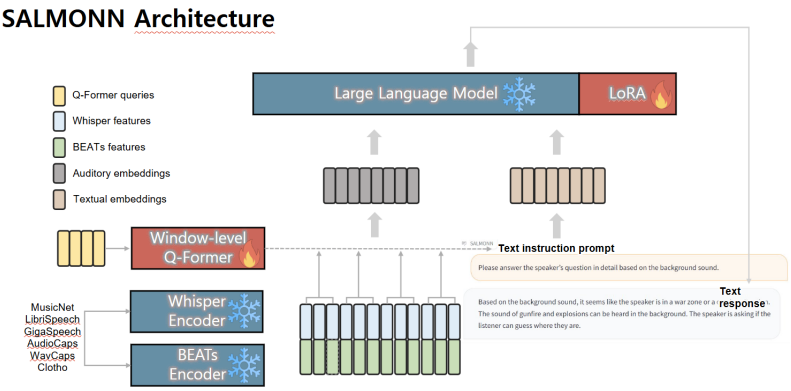

Figure 8. SALMONN 기반 모델 구조 (예시)

6.1 SALMONN 구조

본 프로젝트에서는 ASR, AAC 등 다중 태스크를 처리할 수 있는 SALMONN 모델을 기본 골자로 삼았다. 경량화를 위해 비교적 작은 LLM 백본을 사용하고, Speech/Audio 인코더를 교체해 실험했다.

6.2 LLM Backbone

4B 파라미터 이하의 LLM을 탐색한 뒤, llama 3.2 3B, Gemma 2B, Qwen2.5 3B, phi 3.5-mini, Mistral-7B-GPTQ 등을 후보로 올렸다. 최종적으로 **llama 3.2 3B**가 GPU 사용량 대비 성능이 가장 우수해 채택되었다.

6.3 Speech Encoder

SALMONN은 기본적으로 Whisper Large 모델의 Encoder를 사용한다. 대체 후보로 Moonshine, HuBERT, Seamless-m4t 등을 검토했으나, 시간 문제로 whisper v2, v3, v3-turbo만 실제 실험했다. 가장 성능이 높았던 **whisper large v3-turbo**를 최종 선택했다.

6.4 Audio Encoder

Non-speech 오디오에 특화된 BEATs를 사용한 SALMONN 모델을 베이스로, Ced 인코더도 테스트했으나 성능이 낮았다. 결과적으로 BEATs 인코더를 유지하고 LoRA로 파인튜닝을 시도하였으며, ASR 0.0460 WER, AAC 0.3757 SPIDEr를 달성했다. 메모리 사용량은 평균 9.73GB, Latency는 0.2157sec 수준이었다.

7 Experiments

8 최적화

8.1 서버 아키텍쳐 분석

프로젝트에 사용한 GPU는 V100(구형 아키텍처)로, INT4나 BF16 자료형을 지원하지 않는다. Flash-Attention도 불가능하다. PyTorch 2.0 이상에서 지원하는 SDPA(Scaled Dot Product Attention)를 활용했으나, 최신 GPU 대비 성능 향상이 크지 않았다. 테스트 환경에서는 상대적으로 최신 GPU를 사용할 가능성이 있어, 해당 환경에선 추가적인 성능 이점이 있을 수 있다.

8.2 Liger-Kernel

Liger-Kernel은 Multi-GPU 환경을 위한 Triton Kernel 커뮤니티 포크다. 사용 서버(V100 기반)에서는 성능 향상을 거의 얻지 못했으나, 최신 아키텍처 환경에서 성능 증대가 기대된다.

8.3 Bucketing BatchSampler

오디오 길이가 다양해, 가장 긴 샘플에 맞춰 패딩하면 학습 속도가 저하된다. 오디오 길이 메타데이터를 활용해 비슷한 길이끼리 묶는 방식(Bucketing)을 적용해 **패딩 낭비**를 줄이고 일부 Curriculum Learning 효과도 노렸다.

8.4 Accelerate

분산 학습 라이브러리로, PyTorch DDP보다 설정이 간단해 멀티 GPU 환경을 쉽게 구성할 수 있었다. 서버에서 GPU가 2개만 장착되어 있으나, DDP 설정의 복잡함을 줄이고, 각 GPU 사용량을 조금 더 균형 있게 분산시켜 **효율적인** 학습을 가능케 했다.

8.5 SDPA (Scaled Dot Product Attention)

PyTorch 2.0 이상 버전에서 지원하는 최적화된 Attention 연산이다. SALMONN 내부의 Q-Former, Whisper, BEATs 등 구형 코드 기반 모듈을 수정하여 SDPA를 적용했다. Attention 연산에 대한 연산량이 큰 모델 구조에서 처리가 빨라질 수 있으나, 구형 GPU(V100)에서는 기대만큼의 가속도가 나오지 않았다.

8.6 Torch.compile()

PyTorch 2.0 이상의 동적 그래프를 **정적 그래프**로 변환해 실행 속도를 높일 수 있는 기능이다. C++ 혹은 Triton으로 Wrapping할 필요 없이 간단히 최적화를 적용할 수 있었다. SALMONN처럼 모듈 간 호출이 많은 모델에서 오버헤드를 줄여, 약간의 처리 속도 향상을 확인했다.

8.7 Torch.TensorRT

TensorRT는 NVIDIA의 추론 엔진으로, GPU에서 레이어별 최적화를 수행해 추론 속도를 크게 높일 수 있다. Torch-TensorRT를 이용하면 PyTorch 코드에서 직접 TensorRT 최적화를 적용 가능하다. SALMONN 모델은 여러 하위 모델들이 결합된 구조이긴 하지만, TorchScript 변환 후 TensorRT 최적화 과정을 거쳐 **추론 속도 약 2배** 향상을 확인했다(Compile 시에는 다소 시간이 걸림).

9 Conclusion

오디오 언어모델(Automatic Speech Recognition, Audio Captioning 중심)을 경량화하기 위한 레서피를 제안하였다. 주된 아이디어로는 양자화 등의 모델 차원의 최적화 기법과, 노이즈 고려 전처리(VAD 필터링 등) 같은 데이터 차원 전략을 결합하여, 성능 저하 없이 모델 크기와 계산 비용을 줄이는 방법을 모색하였다. 여러 오디오 데이터셋에서 진행한 실험 결과, 적절한 하드웨어 지원과 데이터 전처리가 뒷받침된다면 제안 기법이 효율적이라는 가능성을 확인할 수 있었다.

한계점(Limitations) 다만, 본 연구에는 몇 가지 제한 사항이 존재한다. 첫째, 양자화 기법에 집중하느라, 프루닝이나 지식 증류(KD) 등 다른 경량화 기법은 충분히 탐색하지 못했다. 특히, 하드웨어 호환성 문제로 일부 양자화 방식(int4, int8)에서 성능을 제대로 평가하지 못한 점이 아쉽다. 둘째, 노이즈가 심하거나 인공적으로 생성된 오디오 데이터를 효과적으로 식별 · 배제하기 위한 방법론이 부족하여, 데이터 품질 통제 측면에서 추가 연구가 필요하다. 마지막으로 최신 GPU 커널 기반의 최적화들은 구형 하드웨어 환경에서 기대만큼 성능을 끌어내지 못하는 한계가 있었다. 향후 연구에서는 프루닝 · KD를 포함한 다양한 모델 최적화 기법을 통합하고, 노이즈 데이터 식별 전략을 고도화하며, 광범위한 하드웨어 환경에서도 일관된 효율 향상을 보일 수 있는 경량화 방안을 모색할 예정이다.

References

- **Accelerate**, hugging face, `https://huggingface.co/docs/accelerate/index`
- **PEFT**, hugging face, `https://huggingface.co/docs/peft/index`
- **TorchTensorRT**, Pytorch, `https://pytorch.org/TensorRT/`
- **Torch.compile**, Pytorch, `https://pytorch.org/tutorials/intermediate/torch_compile_tutorial.html`
- **scaled_dot_product_attention**, Pytorch, `https://pytorch.org/docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html`
- Liger Kernel: Efficient Triton Kernels for LLM Training (2024) `https://arxiv.org/pdf/2410.10989`
- **Beats: Audio pre-training with acoustic tokenizers. (2022)** `https://arxiv.org/abs/2212.09058`
- **Robust speech recognition via large-scale weak supervision. (2023)** `https://arxiv.org/abs/2212.04356`

- **Salmonn: Towards generic hearing abilities for large language models. (2023)** https://arxiv.org/abs/2310.13289
- **LOAE: LLMs with Optimized Audio Encoding (2024)** https://www.isca-archive.org/interspeech_2024/liu24_interspeech.pdf
- **AudioSetCaps: An Enriched Audio-Caption Dataset using Automated Generation Pipeline with Large Audio and Language Models. (2024)** https://arxiv.org/abs/2411.18953
- **Audiocaps: Generating captions for audios in the wild. (2019)** https://aclanthology.org/N19-1011/
- **Large language models as data preprocessors. (2023)** https://arxiv.org/abs/2308.16361
- **Enhancing Audio Classification Through MFCC Feature Extraction and Data Augmentation with CNN and RNN Models. (2024)** https://thesai.org/Publications/ViewPaper?Volume=15&Issue=7&Code=IJACSA&SerialNo=4
- **Audio spectrogram representations for processing with convolutional neural networks. (2017)** https://arxiv.org/abs/1706.09559
- **Wavcaps: A chatgptassisted weakly-labelled audio captioning dataset for audio-language multimodal research. (2024)** https://arxiv.org/abs/2303.17395
- **A survey of data augmentation for audio classification. (2022)** https://www.sba.org.br/cba2022/wp-content/uploads/artigos_cba2022/paper_5085.pdf
- Whisper v3: https://huggingface.co/openai/whisper-large-v3
- Whisper v3 Turbo: https://huggingface.co/openai/whisper-large-v3-turbo
- LOAE: https://github.com/frankenliu/LOAE
- Gemma-2-2b-it: https://huggingface.co/google/gemma-2-2b-it
- Qwen2-Audio: https://github.com/QwenLM/Qwen2-Audio
- Qwen2.5-3B-Instruct: https://huggingface.co/Qwen/Qwen2.5-3B-Instruct
- Mistral-7B-Instruct-v0.3-GPTQ: https://huggingface.co/thesven/Mistral-7B-Instruct-v0.3-GPTQ
- Phi-3.5-mini-instruct: https://huggingface.co/microsoft/Phi-3.5-mini-instruct
- Chen, Sanyuan, et al. "Beats: Audio pre-training with acoustic tokenizers." *arXiv preprint arXiv:2212.09058* (2022)
- Radford, Alec, et al. "Robust speech recognition via large-scale weak supervision." *International conference on machine learning*, PMLR (2023)
- Tang, Changli, et al. "Salmonn: Towards generic hearing abilities for large language models." *arXiv preprint arXiv:2310.13289* (2023)
- Liu, Jizhong, et al. "Enhancing Automated Audio Captioning via Large Language Models with Optimized Audio Encoding." *arXiv preprint arXiv:2406.13275* (2024)

10 개인 회고

10.1 김보현

학습 목표 달성을 위한 노력 오디오 경량화와 최적화 방안을 찾기 위해, **총 6개 데이터셋**을 발굴·정리하고, 허깅페이스에서 제공되는 모델과 해당 논문들을 심도 있게 검토했다. 서치를 바탕으로 오디오의 길이, 무음 구간, 스펙트럼 특징, 잡음 분포 등을 전처리(클리핑 제거, 리샘플링)와 증강 전략을 구체화했다.

마주한 한계와 아쉬웠던 점 토큰 프루닝을 **Vision Transformer** 방식으로 오디오 스펙트로그램에 적용하려 했으나 구현에 어려움을 겪어 결과적으로 적용하지 못했다. 여러 소스의 오디오 데이터(6개 데이터셋)를 분석하면서, 서로 다른 **샘플레이트와 형식**을 맞추는 데 어려움이 있었다. 전처리 규칙(무음 제거, 길이 패딩 등)이 태스크마다 달라, 통일된 파이프라인을 구축하는 데 더 많은 시간이 필요했다.

10.2 김재환

학습 목표 달성을 위한 노력 익숙하지 않은 도메인이기에 기초부터 다시 훑고 관련 논문을 찾아봤다. 특히 SALMONN의 듀얼 인코더 구조를 토대로 **인코더 교체**와 **LLM 백본**을 여러 조합으로 실험했으며, 데이터 증강 등을 통해 모델 성능 개선에 기여하고자 했다.

마주한 한계와 아쉬웠던 점 SALMONN 소형 모델로 **KD(지식 증류)**를 시도했으나, 듀얼 인코더 특성 탓에 완벽한 구현에 실패했다. 또한 Whisper 외의 **스피치 인코더**를 깊이 탐색하거나, 듀얼 인코더가 아니면서 ASR & AAC를 모두 처리하는 모델도 충분히 시험하지 못한 점이 아쉽다.

10.3 김진석

학습 목표 달성을 위한 노력 서버가 한 대뿐이었던 상황에서, 모델 테스트 중간에 코드를 작성하거나 다른 기법을 검토하여 **효율적 자원 활용**을 시도했다. TensorRT, PyTorch Accelerate, Quantization, PEFT, Optimum 등 다양한 최적화 프레임워크를 실험했다.

마주한 한계와 아쉬웠던 점 **하드웨어 한계**로 인해 int4, int8 양자화를 제대로 검증하지 못했고, 모델 구조가 복합적이다 보니 다른 최적화 시나리오도 시도하기 어려웠다. TensorRT만 집중적으로 다룬 것이 아쉬우며, **TensorRT-LLM, Optimum-ONNX** 등도 병행했다면 더 좋은 결과를 얻을 수 있었을 것으로 보인다.

10.4 박진영

학습 목표 달성을 위한 노력 Audio LLM과 멀티모달 개념에 대한 이해가 부족해 **논문**과 **블로그**를 통해 공부했다. SALMONN 모델을 기준으로 Encoder/LLM을 소형 모델로 교체하며 테스트했고, Encoder에 LoRA를 적용해 파인튜닝을 진행했다. 데이터를 정제해 적은 양으로도 높은 성능을 내는 **Data-driven 실험**도 병행했다.

마주한 한계와 아쉬웠던 점 GPU에서 int4, int8 같은 양자화를 지원하지 않아 **성능 비교**를 못했고, GPU 메모리 관리가 미숙해 프루닝된 LLM도 실행하지 못했다. Beats 대신 Ced 인코더를 적용해 보려 했으나, 시간이 부족해 검증을 충분히 못 했다는 점이 아쉽다.

10.5 성기훈

학습 목표 달성을 위한 노력 - 모델, 데이터, 코드 파트를 계속 **소통**하며 팀원들을 독려 - **APOLLO, Liger-Kernel, torch.compile** 등 최신 기법을 일부 적용하여 로우레벨 최적화를 시도 - SALMONN의 **어텐션** 구조를 수정하고, CLAP, Silero VAD 등으로 데이터 품질을 관리

마주한 한계와 아쉬웠던 점 프로젝트 리더로서 **방향성**을 충분히 제시하지 못했고, 하드웨어 한계를 더 일찍 파악하지 못해 최적화 시도에서 어려움을 겪었다. 결과적으로 **구형 아키텍처**에선 개선 효과가 미미해 아쉬움이 남는다.

10.6 양호철

마주한 한계와 아쉬웠던 점 오디오 태스크 관련 **정보 수집**이 원활하지 않아 팀 내 지식 공유가 부족했다. 노이즈나 인공적 데이터 구별 기법을 찾지 못했고, 증강 적용이 충분치 않아 유의미한 성능 향상을 확인하기 어려웠다. 데이터 문제 해결에서 **팀원 요청**이나 추가 방법 모색을 더 적극적으로 해야 했다는 아쉬움이 있다.