[충격 소음원 및 방사소음 예측 모델 연구 보고서]

[작성자 이름]
정영운 / 정준우 / 명근호

[소속]
인하대학교 데이터사이언스학과

[제출일: 2025-09-30]

1 서론 (Introduction)

1.1 문제 정의 (Problem Definition)

본 연구는 녹음된 오디오(.wav) 파일로부터 충격 소음 데이터를 분석하여 두 가지 상이한 과제를 동시에 해결하는 것을 목적으로 한다. 이는 다중 작업 학습**(Multi-Task Learning, MTL)** 문제로 정의된다.

1.1.1 충격원 분류 **(Source Classification):** 소음을 발생시킨 근원(예: desk, chair, hammer 등 5가지 클래스)을 식별하는 작업이며, 다중 클래스 분류 **(Multi-class Classification)** 과제이다.

1.1.2 방사소음 크기 예측 **(Decibel Prediction):** 해당 충격으로 인해 발생한 소음의 실제 크기(dB)를 수치적으로 예측하는 작업이며, 회귀 **(Regression)** 과제이다.

이 두 과제는 모두 오디오 신호의 공통적인 음향 특징을 기반으로 하므로, 단일 모델 구조 내에서 특징 표현을 공유하는 **MTL** 프레임워크가 가장 효율적인 접근 방식이다.

1.2 연구 목표 및 평가 기준 (Research Objectives and Evaluation Criteria)

본 연구의 최종 목표는 실제 환경의 충격 소음 데이터에 대해 높은 일반화 성능을 보이며, 정성적(분류) 및 정량적(회귀) 분석 결과를 동시에 제공할 수 있는 강건한 통합 모델을 구축하는 것이다.

각 작업의 성공적인 완수 여부를 판단하기 위해 다음의 핵심 평가 지표(Key Evaluation Metrics)를 채택한다.

1.2.1 충격원 분류 **(Classification)** 평가 지표**: F1-Score**

- 지표 선정 근거**:** 분류 모델의 성능 평가 시 단순 정확도(Accuracy)는 클래스 불균형에 민감하며 오도될 수 있다. F1-Score는 정밀도(Precision)와 재현율(Recall)의 조화 평균을

사용하여 오탐 및 미탐률을 종합적으로 고려한 성능을 측정함.

- 핵심 목표**:** 모델이 특정 다수 클래스에 편향되지 않고 모든 충격원 클래스에서 고르게 안정적인 성능을 달성하는 것이 중요하므로, Weighted Average F1-Score를 최종 목표 지표로 설정한다.

1.2.2 방사소음 예측 **(Regression)** 평가 지표**: MAE**

- 지표 선정 근거**:** 소음 레벨(dB) 예측은 수치 예측 작업이며, 예측 오차의 크기를 직관적으로 해석할 수 있어야 한다. MAE (Mean Absolute Error, 평균 절대 오차)는 예측 오차를 실제 dB 단위로 직접 표현하며, 오차에 선형적인 패널티를 부여한다.
- 핵심 목표**:** MAE는 이상치(Outlier)의 영향을 최소화하면서 모델의 평균적인 예측 정확도를 가장 잘 반영하는 지표이다. MAE를 최소화하여 실제 소음 크기에 대한 예측 신뢰도를 확보하는 것이 주요 목표이다.

2 모델 설계 **(Model Design)**

2.1 데이터 전처리

- 입력 데이터**:** 원본 .wav 오디오 파일
- 음성 특징 추출 함수 (`make_features`) 분석

본 프로젝트는 오디오 신호를 딥러닝 모델의 입력으로 사용하기 위해 표준화된 특징(Feature) 벡터로 변환하는 과정을 포함한다. 이 변환을 담당하는 핵심 함수가 `make_features`이며, 그 역할은 **Mel-filterbank(**멜 필터뱅크**)** 스펙트로그램을 생성하고, 이를 일정한 길이로 정규화하는 것이다.

이 함수는 `torchaudio` 라이브러리를 사용하여 음성 처리의 표준적인 절차를 따르며, 재현성 및 모델 학습의 안정성을 높였다.

2.1.1 함수 정의 및 입력 인자

인자명	역할	기본값
`wav_name`	처리할 오디오 파일의 경로 (문자열)	-
`mel_bins`	생성할 멜 필터뱅크의 개수 (특징 차원)	-
`target_length`	생성될 특징 맵의 시간 축 길이 (프레임 수)	1024

2.1.2 특징 추출 과정 상세 설명

단계 **1:** 오디오 로드 및 리샘플링

```
waveform, sr = torchaudio.load(wav_name)
if sr != 16000:
    waveform = torchaudio.transforms.Resample(sr, 16000)(waveform)
sr = 16000
```

- `torchaudio.load(wav_name)`: 지정된 경로에서 오디오 파일(`wav_name`)을 로드하여 파형**(waveform)** 데이터와 샘플링 속도(sr)를 얻었다.
- 표준화 **(16kHz)**: 오디오 신호 처리에 있어 표준적인 샘플링 속도인 16,000Hz를 사용하도록 보장합니다. 로드된 `sr`이 16,000Hz가 아닐 경우, `torchaudio.transforms.Resample`을 사용하여 오디오 파형을 16,000Hz로 리샘플링했다. 이는 모든 입력 데이터의 시간 해상도를 통일하여 모델 학습의 일관성을 확보하였다.

단계 **2: Mel-filterbank** 특징 추출

```
fbank = torchaudio.compliance.kaldi.fbank(
    waveform, htk_compat=True, sample_frequency=sr, use_energy=False,
    window_type='hanning', num_mel_bins=mel_bins, dither=0.0,
    frame_shift=10
)
```

- 특징 변환: `torchaudio.compliance.kaldi.fbank` 함수를 사용하여 오디오 파형을 **Mel-filterbank** 스펙트로그램으로 변환했다. 이 함수는 음성 인식 분야에서 널리 사용되는 Kaldi 툴킷과의 호환성을 위해 사용된다.
- 주요 파라미터:
    - `num_mel_bins`: 입력 인자 `mel_bins`에 따라 특징 벡터의 차원을 결정한다.
    - `frame_shift=10`: 프레임 간의 이동 거리(Frame Shift)를 10ms로 설정하며, 이는 초당 100개의 특징 프레임을 생성함을 의미한다.
    - `use_energy=False`: 프레임 에너지(에너지를 특징 벡터에 포함하는 것)를 사용하지 않도록 설정한다.

단계 **3:** 패딩 또는 잘라내기 **(Padding or Truncation)**

```
n_frames = fbank.shape[0]
p = target_length - n_frames
if p > 0: fbank = torch.nn.ZeroPad2d((0, 0, 0, p))(fbank)
elif p < 0: fbank = fbank[0:target_length, :]
```

- 길이 통일: 딥러닝 모델의 입력은 보통 고정된 크기여야 하므로, 특징 맵의 시간축(`n_frames`)을 `target_length`로 맞췄다.
- 패딩 **(Padding)**: `p > 0`일 경우, 원래 길이가 `target_length`보다 짧으므로, 특징 맵의 끝에 0을 채워(`ZeroPad2d`) 길이를 늘렸다.
- 잘라내기 **(Truncation)**: `p < 0`일 경우, 원래 길이가 `target_length`보다 길므로, 특징 맵의 첫 부분(`0`부터 `target_length`까지)만 사용하여 불필요한 뒷부분을 잘라냈다.

단계 **4:** 정규화 **(Normalization)**

```
fbank = (fbank - (-4.2677393)) / (4.5689974 * 2)
```

- 전역 정규화: 생성된 멜 필터뱅크 특징에 사전에 계산된 전역 평균 및 표준편차를 사용하여 정규화를 수행한다. 이 과정은 모델 학습의 안정성과 수렴 속도를 높이는 데 필수적이다.
- 정규화 공식: Normalized Feature=(Feature−μ)/σ′
    - μ=−4.2677393 (사전 계산된 글로벌 평균)
    - σ′=4.5689974×2 (사전 계산된 표준편차에 2배를 곱한 값, 특정 데이터셋의 특성을 고려한 스케일링 요소)

단계 **5:** 예외 처리 및 반환

```
except Exception as e:
    print(f" - '{os.path.basename(wav_name)}' 파일 처리 중 오류가 발생했습니다: {e}")
    return None
```

- `try-except` 블록을 사용하여 파일 로드 또는 특징 추출 과정에서 오류가 발생했을 경우 프로그램을 중단하지 않고, 오류 메시지를 출력한 후 `None`을 반환한다.

2.2 모델 아키텍처

본 프로젝트의 모델인 `MultiTaskAST`는 AST를 기반으로 하며, 단일 백본(Backbone)을 공유하는 다중 작업 학습**(MTL)** 패러다임을 따른다.

2.2.1 AST 백본의 구조 및 특징 공유

- **Backbone** 채택**:** AST는 Vision Transformer (ViT) 구조를 채택하여, 멜 스펙트로그램을 이미지처럼 처리한다. 스펙트로그램 패치들을 시퀀스로 분할하고 Transformer 인코더를 통과시켜 시간적-주파수적 정보를 동시에 효과적으로 인코딩한다.
- 임베딩 추출**:** 오디오 특징이 AST의 다층 Transformer 인코더를 통과한 후, 최종 레이어에서 **[CLS]** 토큰과 **[DIST]** 토큰의 평균을 계산하여 차원의 공통 특징 벡터(Shared Embedding)를 추출했다. 이 벡터는 소리의 종류(충격원)와 크기(데시벨)에 대한 정보를 모두 포함하는 고수준의 표현이다.
- 코드 구현**:** `self.base_ast`에 사전 학습된 AST 모델을 정의하며, `get_embedding(x)` 메서드가 이 768차원 특징 벡터 추출을 담당한다.

2.2.2 다중 작업 헤드 **(Multi-Task Heads)** 설계

공유된 768차원 특징 벡터는 두 개의 독립적이고 기능적으로 분리된 신경망 헤드에 입력된다.

2.2.2.(1) 분류 헤드 **(Classification Head)**

- 역할**:** 5가지 충격원 클래스(desk, chair, lecturestand, book, hammer) 중 하나를 예측하는 분류 작업이다.
- 구조**:** LayerNorm(768) to Linear(512) to ReLU to Dropout(0.3) to Linear(num_Classes=5)로 구성된 다층 퍼셉트론(MLP) 구조입니다. 512차원의 은닉층을 사용하여 복잡한 분류 경계를 학습하며, `Dropout`을 적용하여 과적합을 방지한다.
- 손실 함수**:** `nn.CrossEntropyLoss`를 사용하여 다중 클래스 분류 문제를 최적화한다.

2.2.2.(2) 회귀 헤드 **(Regression Head)**

- 역할**:** 충격에 의한 방사소음의 연속적인 데시벨(dB) 값을 예측하는 회귀 작업이다.
- 구조**:** LayerNorm(768) to Linear(256) to ReLU to Linear(output=1)로 구성된 구조이다. 상대적으로 작은 256차원의 은닉층을 사용하여 데시벨 값이라는 단일 연속 값을 예측한다.

- 손실 함수: `nn.MSELoss` (Mean Squared Error Loss)를 사용하여 예측 값과 실제 데시벨 값 사이의 제곱 오차를 최소화한다.

2.3 다중 작업 최적화 및 학습 설정

- 최종 손실 함수: 두 헤드에서 계산된 손실(LossClassification 및 LossRegression)은 단순 합산되어 최종 손실(Total Loss)을 구성한다.
- **MTL**의 이점: 두 과제를 동시에 학습함으로써, 분류 작업에 유용한 특징(소리의 스펙트럼 패턴)과 회귀 작업에 유용한 특징(소리의 에너지 및 진폭)이 AST 백본에서 서로를 보완하며 학습되어 모델의 일반화 성능이 향상된다.
- 최적화 기법: `torch.optim.Adam` 최적화기와 `torch.cuda.amp`의 혼합 정밀도 학습**(Mixed Precision)**(`autocast`, `GradScaler`)을 적용하여 GPU 연산 속도를 높이고 메모리 사용량을 줄여 효율적인 대규모 학습을 가능하게 했다.
- 경사 누적: `accumulation_steps = 4`를 설정하여 배치 크기를 효과적으로 확장하고 학습 안정성을 높였다.

3 실험 및 학습 과정 **(Experiments & Training)**

3.1 실험 환경

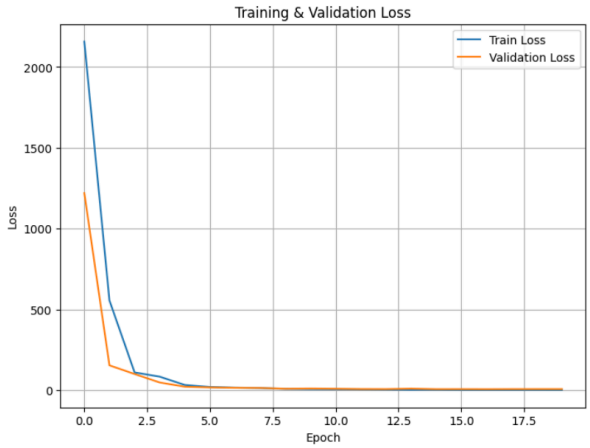
- 하드웨어: Google Colab (GPU: NVIDIA A100)
- 소프트웨어: PyTorch, torchaudio, timm, scikit-learn, pandas
- 주요 하이퍼파라미터:
  - Optimizer: Adam (Learning Rate: 1e-4)
  - Batch Size: 8
  - Gradient Accumulation Steps: 4 (실질 배치 사이즈: 32)
  - Epochs: 20

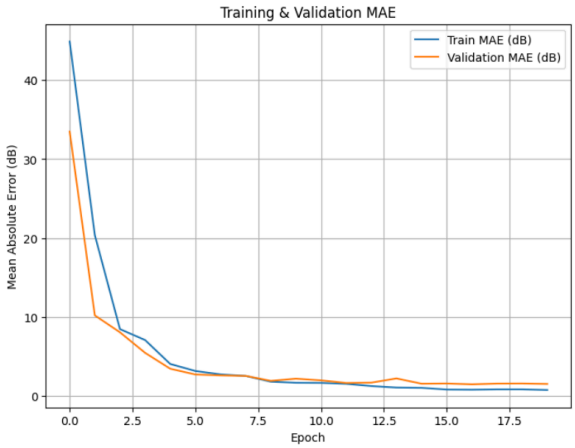
3.2 과적합 방지 전략

- 검증 데이터셋 활용: 훈련 데이터와 별도로 검증 데이터셋을 구성하여 매 에포크 종료 시 모델의 일반화 성능을 평가한다.
- 최적 모델 저장**(Best Model Saving):** 훈련 과정 전체에서 검증 손실**(Validation Loss)**이 가장 낮았던 시점의 모델 가중치를 저장한다**.** 최종 평가에는 이 '최적 모델'을 사용하여 과적합된 모델이 선택될 위험을 배제하였다.

3.3 학습 히스토리

- 아래 그래프는 에포크별 훈련 손실(Train Loss)과 검증 손실(Validation Loss)의 변화를 나타냈다.





- 분석: 그래프에서 볼 수 있듯, 훈련 손실과 검증 손실이 초반에 함께 빠르게 감소한 후 안정적으로 수렴한다. 두 곡선 사이에 큰 간격이 발생하지 않아, 본 모델이 과적합 없이 안정적으로 학습되었음을 확인할 수 있다.

4. 실험 결과 및 성능 (Results & Performance)

(주의사항: 아래 모든 성능 지표는 학습에 일절 사용되지 않은 ***Test*** 세트로 측정한 결과이다.)

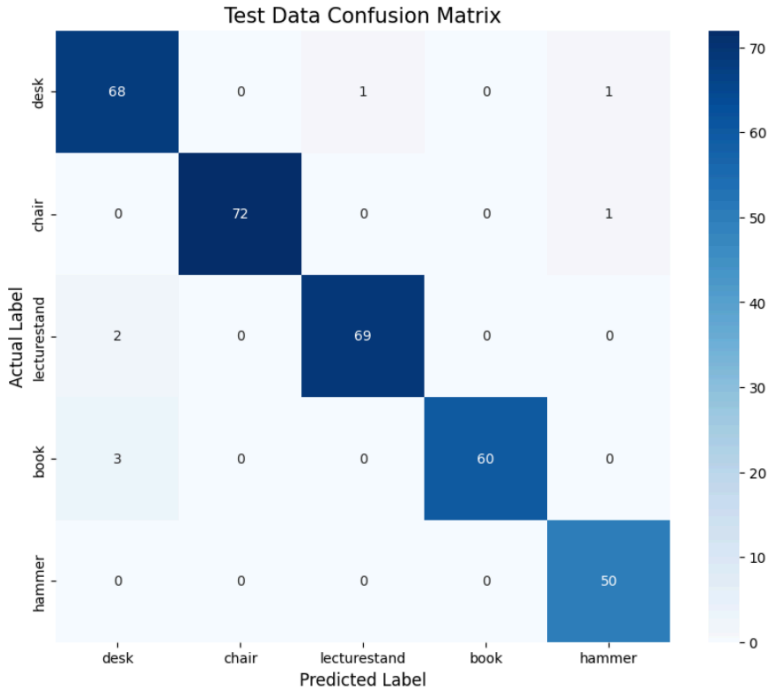
4.1 충격원 분류 성능

- **Classification Report:**

[Classification Report]

	precision	recall	f1-score	support
desk	1.00	0.95	0.98	63
chair	1.00	0.99	0.99	73
lecturestand	0.93	0.97	0.95	70
book	0.96	1.00	0.98	50
hammer	0.99	0.97	0.98	71
accuracy			0.98	327
macro avg	0.98	0.98	0.98	327
weighted avg	0.98	0.98	0.98	327

- **Confusion Matrix:**



- 최종 정확도**(Accuracy): 97.55%**
- 최종 **F1-Score (Weighted Avg): 0.98**

4.2 방사소음 예측 성능

- 최종 평균 절대 오차**(MAE): 1.4121 dB**
- 최종 **R^2 Score: 0.9347**

4.3 일반화 성능 분석 **(**검증 **vs.** 테스트**)**

지표	**Validation Set**	**Test Set**	변화
분류 정확도	97.52%	97.55%	**+0.03%p**
MAE (dB)	1.5021	1.4121	**-0.09**

- 분석**:** 위 표에서 보듯, 모델은 처음 보는 Test 세트에서도 Validation 세트와 대등하거나 오히려 더 향상된 성능을 보인다. 이는 본 모델이 특정 데이터셋에만 치우치지 않고, 새로운 데이터에 대해서도 높은 일반화 성능을 갖추었음을 입증한다.

5 결론 **(Conclusion)**

본 연구는 AST 기반의 다중 작업 학습 모델을 통해 충격 소음원 분류 및 방사소음 예측 문제를 성공적으로 해결하였다. 특히, 검증 데이터 기반의 최적 모델 저장 전략을 통해 과적합을 효과적으로 제어하고 높은 일반화 성능을 달성했다. 최종 평가 결과, 충격원 분류 F1-score **0.98**, 방사소음 예측 MAE **1.4121dB**라는 결과를 얻을 수 있었다.