

WAVE-U-NET을 이용한 음성 노이즈 제거 모델 구현

WAVE U NET 모델 실습

지도교수 : 김선만
발표자 : AINC 202378167김태연
(2025.01.06)



Contents

01

환경 설정

Pytorch 및 필요 라이브러리 설정
GPU 환경 설정

02

데이터셋 구현

AUDIOMNIST 클래스 구현
데이터 전처리 과정

03

WAVE U NET Model

샘플링 레이어와 스킵 연결
RNN CNN 설명

04

모델 학습 과정

데이터 로더 설정과 손실 함수, 훈련 루프 구현

05

실험 결과 분석

훈련 그래프 분석
테스트 손실 평가와 모델 성능 평가

06

결론

모델 성능 분석과 구현 성과



데이터셋 구현

환경 설정 코드 설명

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchaudio
from torchaudio.datasets import
SPEECHCOMMANDS
import matplotlib.pyplot as plt
import os

print(torchaudio.list_audio_backends
())
print(torch.cuda.is_available())
device = torch.device("cuda" if
torch.cuda.is_available() else
"cpu")
print(f"Using device: {device}")
```

- TORCHAUDIO를 사용하여 음성 데이터 처리
- CUDA 사용 가능 여부 확인
- GPU 사용이 가능한 경우 GPU를, 아닌 경우 CPU를 사용하도록 설정



데이터셋 구현

AUDIOMNIST 데이터셋 코드 설명

```
class AudioMNIST(SPEECHCOMMANDS):
    def __init__(self, root,
download=False):
        super().__init__(root,
download=download)
        self._walker = [
            item for item in
self._walker
            if any(digit in item
for digit in ['zero', 'one', 'two',
'three', 'four', 'five', 'six',
'seven', 'eight', 'nine'])
        ]

        data_dir =
os.path.join(os.getcwd(), "data")
        os.makedirs(data_dir,
exist_ok=True)
        dataset = AudioMNIST(data_dir,
download=True)
```

- SPEECHCOMMANDS 데이터셋을 상속받아 AudioMNIST 클래스 구현
- 숫자 발음에 해당하는 음성 데이터만 선택
- 데이터 디렉토리 생성 및 데이터셋 다운로드



SPEECHCOMMANDS DATASET

SPEECHCOMMANDS DATASET 설명

- 데이터셋 구성 -> 65000개
- 사용한 단어 -> 0~9 총 10개
- 각 데이터 포인트는 파형(오디오 신호), 샘플 레이트, 발화(레이블), 화자 ID, 발화 번호로 구성



데이터셋 구현

전처리 코드 설명

```
def preprocess(waveform,
sample_rate):
    if isinstance(sample_rate,
torch.Tensor):
        sample_rate =
sample_rate[0].item()
        if sample_rate != 16000:
            waveform =
torchaudio.transforms.Resample(sampl
e_rate, 16000)(waveform)
            target_length = 16000
            if waveform.size(-1) <
target_length:
                waveform =
torch.nn.functional.pad(waveform,
(0, target_length -
waveform.size(-1)))
            else:
                waveform = waveform[...
:target_length]
        return waveform
```

- 샘플링 레이트를 16kHz로 표준화
- 모든 음성을 16,000 샘플 길이로 통일
- 필요한 경우 패딩 적용



WAVE U NET 모델 구조

다운샘플링 코드 설명

```
class WaveUNet(nn.Module):
    def __init__(self,
num_layers=4,
num_initial_filters=8):
        super(WaveUNet,
self).__init__()
        self.num_layers =
num_layers
        self.down_convs =
nn.ModuleList()
        for i in range(num_layers):
            in_channels = 1 if i ==
0 else num_initial_filters * (2**i)
            out_channels =
num_initial_filters * (2**(i+1))

self.down_convs.append(nn.Conv1d(in_
channels, out_channels,
kernel_size=15, stride=1,
padding=7))
```

- 4개의 다운샘플링 레이어로 구성
- 각 레이어마다 필터 수가 2배씩 증가
- 커널 크기 15, 패딩 7 사용



WAVE U NET 모델 구조

업샘플링 코드 설명

```
self.up_convs =  
nn.ModuleList()  
for i in range(num_layers,  
0, -1):  
    in_channels =  
num_initial_filters * (2**i) * 2  
    out_channels =  
num_initial_filters * (2**(i-1))  
  
self.up_convs.append(nn.Conv1d(in_ch  
annels, out_channels, kernel_size=5,  
stride=1, padding=2))  
  
self.final_conv =  
nn.Conv1d(num_initial_filters, 1,  
kernel_size=1, stride=1)
```

- 4개의 업샘플링 레이어로 구성
- 스킵 연결을 통해 다운샘플링 특징과 결합
- 커널 크기 5, 패딩 2 사용
- 최종 출력을 위한 1x1 합성곱 사용



스킵 연결

스킵 연결 설명

- 이전 레이어의 정보를 후속 레이어로 직접 전달
- 주로 덧셈이나 연결 방식으로 구현
- 정보 보존과 모델 성능 향상



모델 학습 과정

학습 과정 코드 설명

```
num_epochs = 10
train_losses = []

for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    for i, (waveform, label,
speaker_id, utterance_number) in
enumerate(train_loader):
        waveform =
torch.stack(waveform).to(device)
        if waveform.dim() == 2:
            waveform =
waveform.unsqueeze(1)

        # 노이즈 추가
        noise =
torch.randn_like(waveform) * 0.1
        noisy_waveform = waveform +
noise

        # 모델 학습
        optimizer.zero_grad()
        outputs =
model(noisy_waveform)
        loss = criterion(outputs,
waveform)
        loss.backward()
        optimizer.step()
```

- 10개의 에폭 동안 모델 학습 진행
- 각 배치마다 음성 데이터에 노이즈를 추가
- MSE Loss를 사용하여 원본 음성과 노이즈 제거 결과 비교
- Adam 옵티마이저를 사용하여 모델 파라미터 업데이트



손실 함수와 최적화

코드 설명

```
criterion = nn.MSELoss()  
optimizer =  
optim.Adam(model.parameters(),  
lr=0.001)
```

- MSE Loss: 원본 신호와 출력 신호의 차이를 계산
- Learning rate 0.001로 설정된 Adam 옵티마이저 사용



ADAM 옵티마이저

ADAM 설명

- 딥러닝 모델 훈련을 위한 효율적인 최적화 알고리즘
- 1차 모멘트인 평균과 2차 모멘트인 분산 계산
- 각 매개변수의 학습률 조정 -> 학습률은 반복에서의 스텝 크기



실험 결과 분석

훈련 결과 분석

```
plt.figure(figsize=(10, 5))
plt.plot(range(1, num_epochs+1),
train_losses, marker='o')
plt.title('Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid(True)
plt.savefig('training_loss.png')
plt.close()
print("그래프가 'training_loss.png'
파일로 저장되었습니다.")

def test_model(model, test_loader):
    model.eval()
    test_loss = 0.0
    with torch.no_grad():
        for waveform, label,
speaker_id, utterance_number in
test_loader:
            waveform =
torch.stack(waveform).to(device)
            noise =
torch.randn_like(waveform) * 0.1
            noisy_waveform =
waveform + noise
            outputs =
model(noisy_waveform)
            loss =
criterion(outputs, waveform)
            test_loss +=
loss.item()
    avg_loss = test_loss /
len(test_loader)
    return avg_loss
```

- 10개의 에폭 동안의 손실값 변화를 그래프로 시각화



실험 결과 분석

테스트 결과 분석

```
plt.figure(figsize=(10, 5))
plt.plot(range(1, num_epochs+1),
train_losses, marker='o')
plt.title('Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid(True)
plt.savefig('training_loss.png')
plt.close()
print("그래프가 'training_loss.png'
파일로 저장되었습니다.")

def test_model(model, test_loader):
    model.eval()
    test_loss = 0.0
    with torch.no_grad():
        for waveform, label,
speaker_id, utterance_number in
test_loader:
            waveform =
torch.stack(waveform).to(device)
            noise =
torch.randn_like(waveform) * 0.1
            noisy_waveform =
waveform + noise
            outputs =
model(noisy_waveform)
            loss =
criterion(outputs, waveform)
            test_loss +=
loss.item()
    avg_loss = test_loss /
len(test_loader)
    return avg_loss
```

- 테스트 데이터셋에 대한 평균 손실값 계산
- 노이즈가 추가된 테스트 데이터에 대한 모델의 성능 평가
- 훈련 과정과 동일한 노이즈 레벨(0.1) 적용



코드 실행 결과

결과창 출력

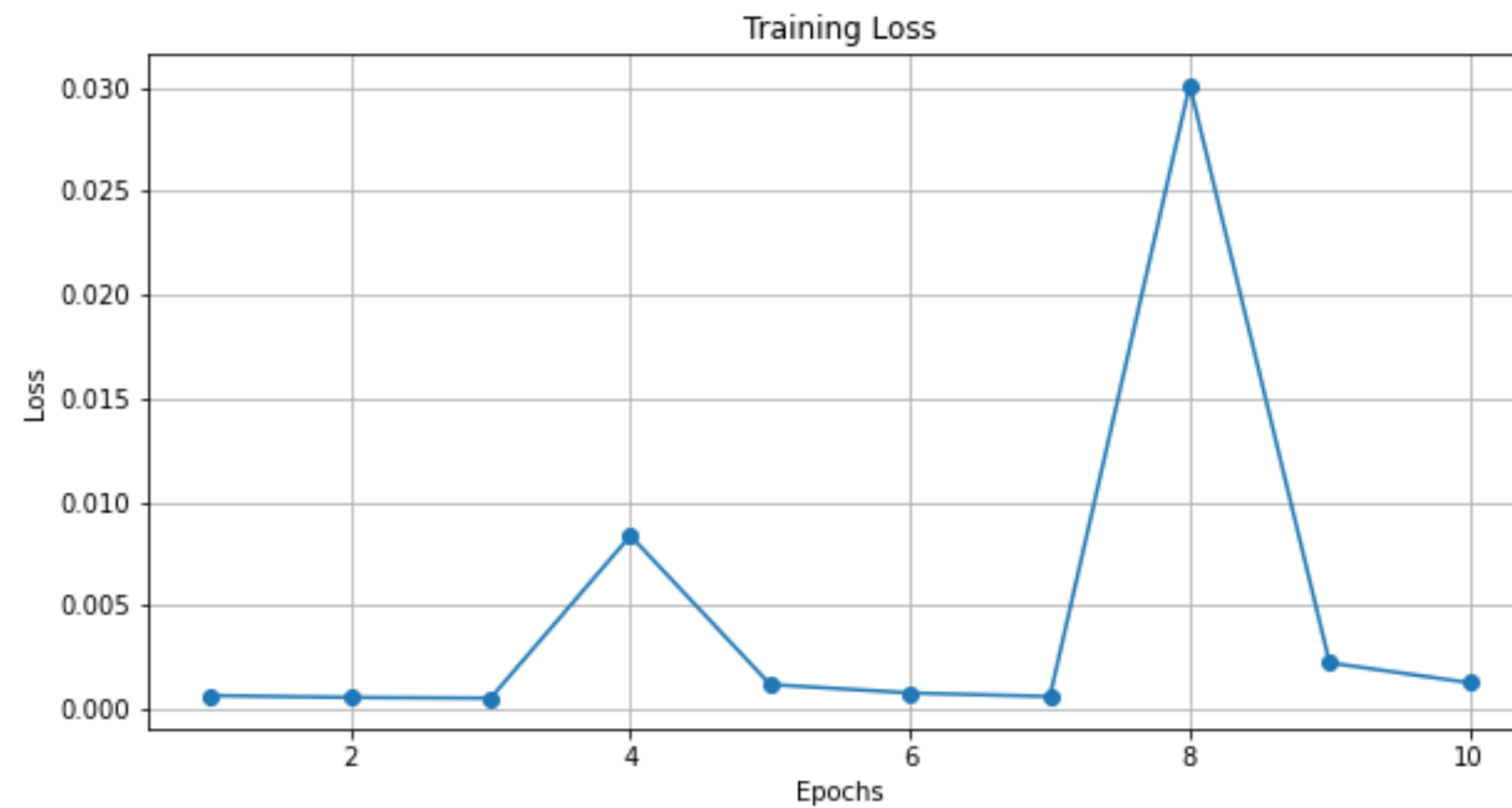
```
['soundfile']  
True  
Using device: cuda  
Epoch [1/10], Loss: 0.0007  
Epoch [2/10], Loss: 0.0006  
Epoch [3/10], Loss: 0.0005  
Epoch [4/10], Loss: 0.0084  
Epoch [5/10], Loss: 0.0012  
Epoch [6/10], Loss: 0.0008  
Epoch [7/10], Loss: 0.0006  
Epoch [8/10], Loss: 0.0301  
Epoch [9/10], Loss: 0.0022  
Epoch [10/10], Loss: 0.0013  
그래프가 'training_loss.png' 파일로 저장되었습니다.
```

- 일부 에폭에서 손실값의 급격한 변동이 관찰됨
- 손실값이 전체적으로 감소하는 경향으로 보임



손실값 그래프

그래프 이미지



결론

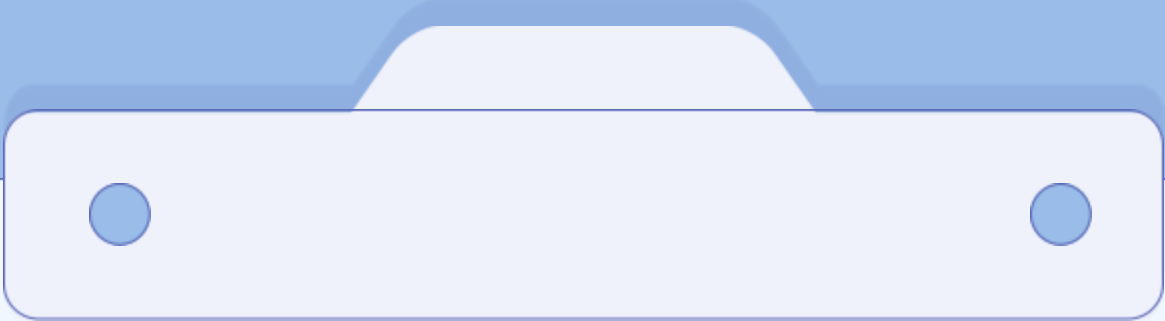
구현 성과 및 향후 개발 방향

```
# 모델 저장
torch.save(model.state_dict(),
'wave_unet_model.pth')
print("모델이 'wave_unet_model.pth'
파일로 저장되었습니다.")

# 테스트 결과
test_loss = test_model(model,
test_loader)
print(f"Test Loss:
{test_loss:.4f}")
```

- 더 다양한 노이즈 패턴에 대한 학습 필요
- 실시간 처리를 위한 모델 경량화 고려
- 다양한 음성 명령어에 대한 테스트 확장





감사합니다

Q & A

지도교수 : 김선만
발표자 : 홍길동

