2023년도 졸업논문

AI 기반의 수면장애 진단 및 방지 기술

합성곱신경망과 모델 경량화 기술을 통한 Embedded AI 구현

한국공학대학교

전자공학부

박준영, 문동주, 유인호

AI 기반의 수면장애 진단 및 방지 기술

합성곱신경망과 모델 경량화 기술을 통한 Embedded AI 구현

AI-based Sleep Disorder Diagnosis and Prevention Technology

2023年　11月　30日

한국공학대학교

전자공학부

박준영, 문동주, 유인호

AI 기반의 수면장애 진단 및 방지 기술

指導敎授 김 평 수 (인)

이 論文을 졸업논문으로 提出함.

2023年 11月 30日

한국공학대학교

전자공학부

박준영, 문동주, 유인호

2023년도

졸업논문

AI 기반의 수면장애 진단 및 방지 기술

박준영 문동주 유인호

<목 차>

그림목차

요약

AI 기반의 수면장애 진단 및 방지 기술

韓 國 工 學 大 學 校
電 子 工 學 科
朴 峻 永, 文 東 珠, 劉 印 鎬

대한민국은 OECD 국가 중 대표적인 수면 부족 국가이고, 수면장애 환자가 증가하고 있다. 수면의 질은 사람의 신체적 건강, 정신적 건강, 생명 및 삶의 질에 밀접한 연관이 있다. 수면의 중요성에 대한 사람들의 관심과 수면의 질을 개선하고자 하는 슬립테크 시장의 규모가 가파르게 증가하고 있다. 그리고 최근 비약적인 반도체의 발달로 인해 저전력 MPU에서도 AI 추론이 가능해졌다. AI는 데이터를 기반으로 복잡한 문제를 빠르고 정확하게 해결하는 장점이 있다. 그리고 임베디드시스템은 소형 및 경량화에 따른 휴대성, 경제성, 신뢰성이 높은 장점이 있다. AI와 임베디드시스템의 장점을 합친 On-device 형태의 임베디드 AI로 수면의 질을 향상하는 기술 및 제품을 개발한다면, 규모가 커지고 있는 슬립테크 시장에서 높은 상품성과 경쟁력을 갖출 수 있을 것으로 기대된다. 그리고 건강, 생명, 삶의 질에 밀접한 연관이 있는 수면장애를 개선함으로써 개인뿐만 아니라 사회적으로도 많은 이로움이 있을 것으로 예상된다. 따라서 본 연구에서 목적은 On-device 형태의 임베디드 AI로 수면장애를 진단 및 방지하여 수면의 질을 향상하는 것이다.

본 논문에서 주요한 특징은 이미지데이터와 음성데이터로 수면 자세와 수면장애를 추론하는 높은 성능의 모델을 학습하는 것이다. 그리고 학습된 모델의 파라미터를 양자화하는 모델 경량화 기술을 이용해서 계산 복잡도와 지연시간을 줄이고, 저전력 MPU에서 실시간 추론이 가능하게 한다. 그리고 MCU가 모델의 예측값을 이용하여 스피커, 서보모터, 진동모터를 제어하는 회로를 설계한다. 따라서 실시간 데이터를 이용해서 수면장애를 추론 및 방지하고 수면 자세를 교정해서, 수면의 질을 향상시키는 기술 및 시스템을 개발하는 것이 목표이다.

Key words
단시간 푸리에 변환, 딥러닝, 합성곱신경망, 전이학습, 미세조정, VGG16, 모델 경량화, On-device AI

1. 서론

1.1 연구 배경 및 필요성

대한민국은 OECD 국가 중 대표적인 수면 부족 국가이고, 수면장애 환자가 증가하고 있다. 2016년 OECD 통계 자료에 따르면 한국인의 평균 수면 시간은 하루 7시간 41분이고, 이는 OECD 평균인 8시간 22분보다 41분(약 9%) 부족한 수치이며, OECD 국가 가운데 최하위로 나타났다[1]. 그리고 건강보험심사평가원 자료에 의하면 수면장애 환자가 2015년 약 456,000명에서 2019년 637,000명으로 4년 새 약 39.7%가 증가했으며, 그중 수면 무호흡증 환자는 2015년 약 29,000명에서 2019년 84,000명으로 4년 새 약 1.9배가 급격히 증가했다[1]. 그리고 시장 조사 전문 기업인 엠브레인 트렌드모니터가 전국의 만 15세~64세 남녀 1,000명을 대상으로 수면의 질에 대한 인식 조사를 한 결과에 의하면, 전체의 32.6%가 수면 시간이 부족하다고 느끼는 것으로 나타났다[1].

수면의 질은 사람의 신체적 건강, 정신적 건강, 생명 및 삶의 질에 밀접한 연관이 있다. 수면 부족 또는 수면장애로 인한 낮은 수면의 질은 심혈관 질환, 퇴행성 뇌 질환, 고혈압, 당뇨병 등 여러 가지 신체적 질환뿐만 아니라, 우울증과 같은 정신적 건강과도 높은 상관관계가 있으며, 전반적인 삶의 만족도를 떨어뜨리는 것으로 나타났다[2-3]. 그리고 수면 시간이 6시간 미만인 사람은 수면 시간이 7~9시간 미만인 사람보다 사망 위험이 13%가 높았다[4].

수면의 중요성에 대한 사람들의 관심과 수면의 질을 개선하고자 하는 슬립테크 시장의 규모가 가파르게 증가하고 있다. 국내 슬립테크 시장의 규모는 2011년 4,800억원에서 2021년 약 3조원으로 10년 새 약 5배 이상 증가했으며, 글로벌 슬립테크 시장의 규모는 2026년에 약 40조원까지 성장할 것으로 예측된다[5]. 따라서 슬립테크 기술은 상업적으로도 가치가 높으며, 많은 사람의 수면의 질을 향상하므로 개인적인 문제 해결뿐만 아니라 사회적으로도 많은 이로움이 있을 것으로 예상된다.

최근, 비약적인 반도체의 발달로 인해 저전력 MPU에서도 AI 추론이 가능해졌다. 그리고 ‘TensorFlow Lite’, ‘PyTorch Mobile’ 등 딥러닝프레임워크의 모델 경량화 라이브러리를 이용하면, 리소스가 제한적인 MCU 및 임베디드 기기에서도 학습된 모델을 이용해서 추론하는 것이 쉬워진다. AI는 데이터를 기반으로 복잡한 문제를 빠르고 정확하게 해결하는 장점이 있고, 임베디드시스템은 소형 및 경량화에 따른 휴대성, 경제성, 신뢰성이 높은 장점이 있다.

AI와 임베디드시스템의 장점을 합친 임베디드AI로 수면의 질을 향상하는 기술 및 제품을 개발한다면 규모가 커지고 있는 슬립테크 시장에서 높은 상품성과 경쟁력을 갖출 수 있으며, 건강과 생명, 삶의 질에 밀접한 연관이 있는 수면장애를 개선함으로써 개인뿐만 아니라 사회적으로도 많은 이로움이 있을 것으로 예상된다. 따라서 저전력 MCU로 수면장애를 추론 및 방지하는 On-device 형태의 임베디드AI 기술을 연구하고자 한다.

1.2 연구 목표

수면의 질은 사람의 건강, 생명, 삶의 질에 밀접한 연관이 있으며, 수면의 중요성에 대한 사람들의 관심과 슬립테크 시장의 규모가 가파르게 증가하고 있다. 그리고 최근 비약적인 반도체의 발달로 인해 저전력 MPU에서도 AI 추론이 가능해졌다. 따라서 본 연구에서는 실시간 데이터를 이용해서 수면장애를 진단 및 방지하고, 수면 자세를 교정해서 수면의 질을 향상시키는 On-device AI를 개발하고자 한다.

본 연구의 목표를 구체화하면 다음과 같다. 첫째, 딥러닝을 활용한 Multi Classification CNN 알고리즘으로 수면 자세와 수면장애를 추론하는 높은 성능의 모델을 학습한다. 둘째, 모델의 복잡도와 연산량을 줄여서 저전력 MPU에서 실시간 데이터로 추론하는데 걸리는 지연시간을 최소화한다. 셋째, 모델이 추론한 결과를 이용해서 수면 자세를 교정하고 수면장애를 방지하는 시스템을 설계한다.

2. 해결 과정

2.1 이미지데이터로 수면 자세를 추론하는 모델 설계 및 학습

4가지 클래스로 구성된 3차원 배열로 이루어진 이미지데이터 총 1,528개와 딥러닝프레임워크 PyTorch를 이용해서 지도학습방식으로 Multi Classification CNN 모델을 학습한다. 수면 자세 데이터는 보건의료 데이터로 분류되어 있어 데이터를 외부에서 수집 및 활용하는 데 어려움이 있었고, 이 문제를 해결하고자 직접 데이터를 생성하였다. 데이터 편향 문제를 방지하고자, 카메라의 각도, 피사체까지의 거리, 성별, 조명 세기, 침구 주변 환경 등을 다양하게 고려해서 데이터를 생성하였다. 데이터 분포는 '수면 자세가 정자세인 경우', '왼쪽으로 치우쳐진 경우', '오른쪽으로 치우쳐진 경우', '사람이 침대에 없는 상황' 4가지 클래스로 구분되며, 모델이 특정 클래스에 과대적합 되는것을 방지하기 위해 동일한 비율로 구성했다. 그리고 클래스 비율을 유지하면서 무작위로 추출하여 Train Data Set, Validation Data Set, Test Data Set이 약 7:2:1 비율이 되도록 분할하였고, 모델 성능 검증 단계에서 과대적합 발생 유무를 좀 더 상세하게 분석하고자 검증 데이터의 비율을 다소 높게 설정하였다.

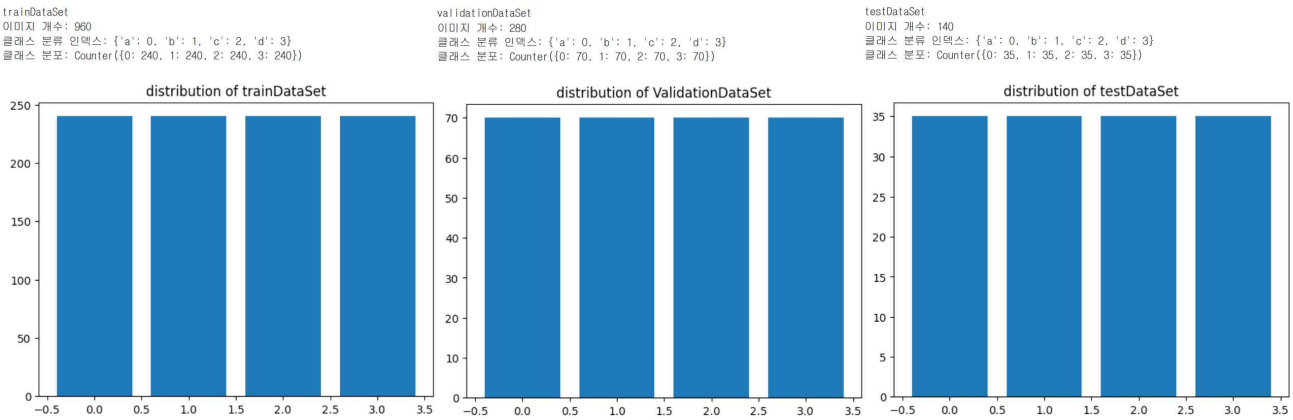

그림 2.1 데이터 분할 및 클래스별 분포(데이터 분할 비율: 7:2:1)

데이터 수집, 데이터 전처리, 모델 설계 및 학습 과정에서 다양한 기법들로 과소적합 및 과대적합 문제를 방지하고, 하이퍼파라미터를 바꿔가며 모델 성능을 검증 및 비교하여 일반화 성능이 높은 모델을 학습한다. 훈련데이터가 부족해서 발생하는 과소적합 문제를 방지하고자, VGG16 전이학습모델을 이용하여 대량의 데이터로 사전 학습된 파라미터값을 훈련데이터로 재학습하는 전이학습(Transfer Learning) 방법으로 모델을 설계 및 학습하였다. 그리고 Bias Error는 작지만, Variance Error가 커서 Total Error가 커지는 과대적합 문제를 방지하고자 다양한 기법을 적용하였다. 데이터 전처리 단계에서 검증 데이터 비율을 다소 높게 분할하여 과대적합 문제를 정밀하게 탐지 및 분석하고자 하였고, 모델 설계 및 학습 과정에서는 모델의 복잡도와 분류하고자 하는 클래스 개수를 고려해서 모델의 FC(Fully Connected Layer)를 수정하였고, 랜

덤하게 일부 노드의 가중치를 0으로 계산해서 최적화하는 Drop out 기법을 모델의 FC(Fully Connected Layer)에 적용하고, 모델 앞단 일부 Layer의 파라미터를 동결시키고 학습하는 미세조정 기법을 이용하고, Train Data Set에 Random Crop을 적용하고, MBGD(Mini-Batch Gradient Descent) 방법으로 Batch size를 2로 작게 설정해서 regularization과 noise를 증가시켜 모델의 복잡도를 낮추는 방식으로 최적해를 찾도록 하였고, 단위 Epoch마다 Validation Data Set에 대한 정확도를 계산해서 검증 정확도가 이전 Epoch 대비 낮은 경우에는 파라미터를 업데이트하지 않는 방식으로 학습해서 모델이 훈련데이터에 Overfitting 되지 않도록 하였다. 그리고 학습이 끝난 모델에 대해서는 Validation Data Set을 이용해서 클래스별 f1 Score가 균일한지 클래스별 모델의 성능을 분석해서 과소적합된 클래스를 찾고, 필요시 해당 클래스의 데이터를 추가 수집하고 재학습하여 모델 성능을 향상하고자 했다. 그리고 하이퍼파라미터 튜닝을 통해 모델 성능을 비교해서 일반화 성능이 가장 높은 모델을 결정하였다. 최적의 하이퍼파라미터는 교차 엔트로피 오차(Cross Entropy Error, CEE)로 비용함수를 설계하고, Momentum과 RMSprop을 이용한 경사하강법인 Adam 최적화기법을 사용하고, 비선형 활성화 함수는 Relu 함수를 사용해서 학습했을 때 일반화 성능이 가장 높았다. 최종 모델의 일반화 성능은 Test Data Set에 대한 정확도를 지표로 사용했고 94% 성능을 보였다. 그리고 최종 모델을 라즈베리파이에서 이용하기 위해, 모델의 구조와 파라미터를 저장하였다.

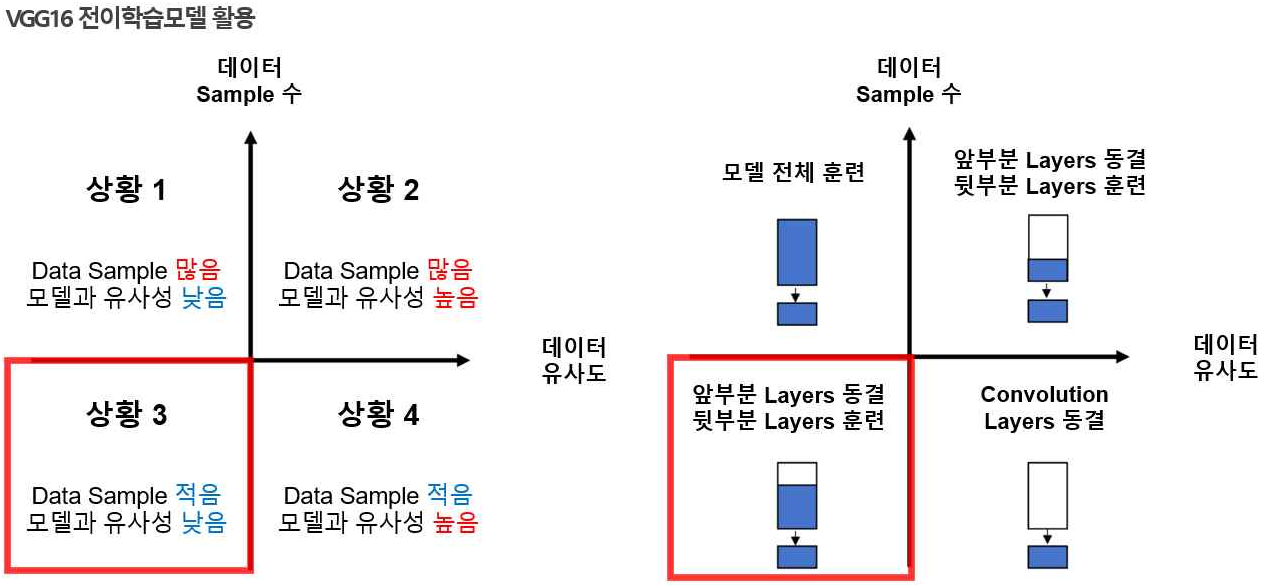

그림 2.2 전이학습모델을 미세조정하여 모델 학습

▾ vgg16 모델 전이학습

```
import torch
import torch.nn as nn
from torchvision.models.vgg import vgg16

# GPU 혹은 CPU 장치 확인
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

model_ft = vgg16(pretrained=True) #vgg 전이학습, 학습된 파라미터 불러옴

fc = nn.Sequential(
    nn.Linear(512 * 7 * 7, 128), # vgg 마지막 CV Layer단 채널수 512 * 7 * 7
    nn.ReLU(),
    nn.Dropout(), # 랜덤하게 일부 노드 가중치를 0으로 하고, 최적화하여 과대적합 방지
    nn.Linear(128, 32),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(32, 4), # 분류 할 클래스 개수
)

model_ft.classifier = fc # vgg 모델 fc단 재설정
model_ft.to(device)
```

▾ vgg16 미세조정

```
for param in model_ft.features[0:].parameters() :
      param.requires_grad = True # 역전파 할 때, 모든 파라미터를 계산 및 업데이트o

for param in model_ft.features[:5].parameters() :
      param.requires_grad = False # 역전파 중, 앞단 일부 파라미터 변화 계산 및 업데이트x
```

그림 2.3 재설계한 FC 구조, 미세조정

▾ 모델 학습

```
num_epochs = 30
criterion = torch.nn.CrossEntropyLoss()
optimizer_ft = optim.Adam(model_ft.parameters(), lr=0.00001)
since = time.time()

best_model_wts = copy.deepcopy(model_ft.state_dict())
best_acc = 0.0

epoch_list, loss_list, acc_list = [], [], []

for epoch in range(1, num_epochs+1):
  print("₩n₩nEpoch {}/{}".format(epoch, num_epochs))
  print("-" * 20)

  for phase in ["train", "val"]:
    if phase == "train":
      model_ft
      model_ft.train()
    else:
      model_ft.eval()

    running_loss = 0.0
    running_corrects = 0

    for inputs, labels in dataloaders[phase]:
      inputs = inputs.to(device)
      labels = labels.to(device)

      optimizer_ft.zero_grad()

      with torch.set_grad_enabled(phase == "train"):
        outputs = model_ft(input  s)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

        if phase == "train":
          loss.backward()
          optimizer_ft.step()

      running_loss += loss.item() * inputs.size(0)
      running_corrects += torch.sum(preds == labels.data)

    epoch_loss = running_loss / dataset_sizes[phase]
    epoch_acc = running_corrects.double() / dataset_sizes[phase]

    epoch_list.append(int(epoch)), loss_list.append(float(epoch_loss)), acc_list.append(float(epoch_acc))

    print("{}₩t Loss: {:.4f}₩t Acc: {:.4f}".format(
        phase, epoch_loss, epoch_acc))

    if phase == "val" and epoch_acc >= best_acc:
      best_acc = epoch_acc
      best_model_wts = copy.deepcopy(model_ft.state_dict())

time_elapsed = time.time() - since
print("Training complete in {:.0f}m {:.0f}s".format(
    time_elapsed // 60, time_elapsed % 60))
print("Best val Acc: {:4f}".format(best_acc))

model_ft.load_state_dict(best_model_wts)
```

그림 2.4 최종 모델의 하이퍼파라미터와 과대적합을 방지하는 학습 구조

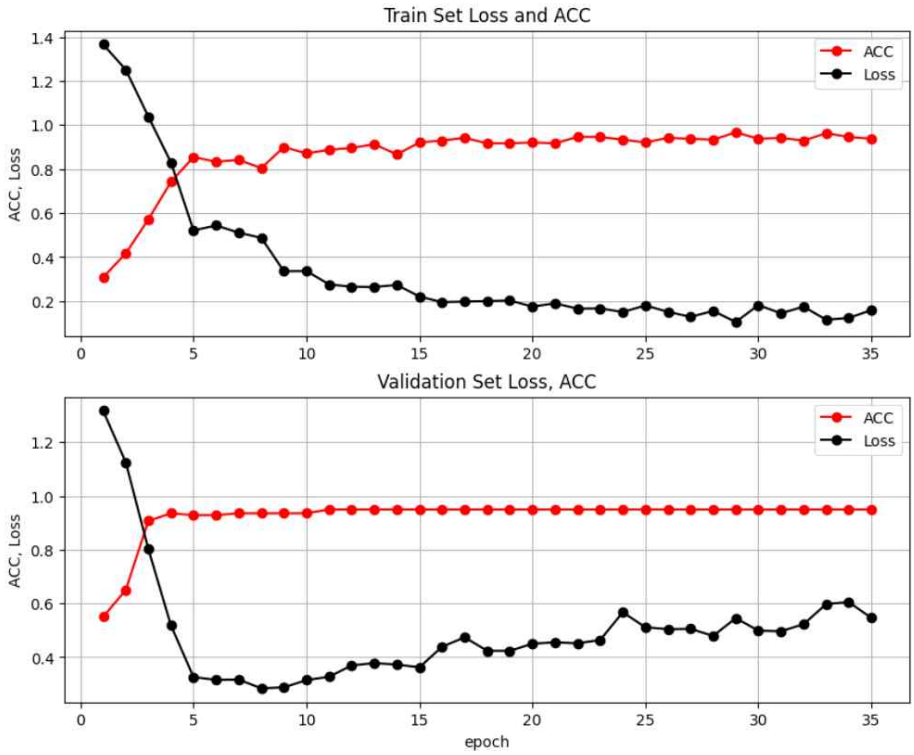

그림 2.5 과대적합 방지를 적용한 모델의 훈련 및 검증 성능

모델 일반화 성능 평가 및 모델 저장

```
def test():
    a = []
    b = []
    model_ft.eval()  # 신경망을 추론 모드로 전환
    correct = 0

    # 데이터로더에서 미니배치를 하나씩 꺼내 추론을 수행
    with torch.no_grad():  # 추론 과정에는 미분이 필요없음
        for data, targets in dataloaders["test"]:
            data = data.to(device)
            targets = targets.to(device)
            outputs = model_ft(data)  # 데이터를 입력하고 출력을 계산

            # 추론 계산
            # 확률이 가장 높은 클래스를 반환
            _, predicted = torch.max(outputs.data, 1)

            # 정답과 일치한 경우 정답 카운트를 증가
            correct += predicted.eq(targets.data.view_as(predicted)).sum()

    # 정확도 출력
    data_num = len(dataloaders["test"].dataset)  # 데이터 총 건수
    print('₩n테스트 데이터에서 예측 정확도: {}/{} ({:.0f}%)₩n'.format(correct,
                                        data_num, 100. * correct / data_num))
test()

# 최종 모델 저장
torch.save(model_ft.state_dict(), 'model11_weights.pth')
```

```
테스트 데이터에서 예측 정확도: 132/140 (94%)
```

그림 2.6 최종 모델의 일반화 성능 및 모델 저장

2.2 음성데이터로 수면장애를 추론하는 모델 설계 및 학습

3가지 클래스로 구성된 3차원 배열로 이루어진 데이터 총 973개를 이용해서 지도학습방식으로 Multi Classification CNN 모델을 학습한다. 모델을 학습하기 위해 수면 음성데이터를 수집하고, labeling 작업을 하고, STFT 기법을 이용해서 시간, 주파수, 신호의 크기 정보가 있는 3차원 배열의 데이터로 전처리한다. 수면 음성데이터는 보건의료 데이터로 분류되어 있어 외부에서 수집 및 활용하는 데 어려움이 있었고, 이 문제를 해결하고자 직접 데이터를 생성하였다. 그리고 데이터 편향으로 발생하는 과대적합 문제를 방지하고자 마이크, 성별, 녹음 환경을 다르게 해서 수면 음성데이터를 녹음하였다. 녹음한 오디오 파일 형식은 표본화 주파수(sampling rate), 양자화 비트 수(Quantization Bit Depth), 마이크의 채널 수를 각각 44,100[Hz], 16[bits], 1[channel]로 하여 녹음 및 데이터 수집을 하였다. 그리고 사람의 평균 호흡 주기는 3~4초이므로[6] margin을 추가하여, 녹음한 수면 음성데이터를 5초 간격으로 분할하고 각각의 데이터에 대해 labeling 작업을 했다. 그리고 데이터를 합성곱신경망 모델에 적합한 입력데이터로 가공하기 위해 STFT 기법으로 속성을 분할해서 3차원 배열의 시계열 데이터로 전처리하였다. STFT 하이퍼파라미터인 window_length와 n_FFT 값은 (sampling rate / 2) 크기로 하였고, window_length로 분할하면 함수의 양쪽 끝에서 불연속점이 발생하므로 Hann window를 적용하여 양쪽 끝이 0이 되도록 하여 연속성을 보장했고, 주파수 해상도를 높이기 위해 hop_length 값은 (window length / 4) 크기로 해서 3/8[s] 주기로 1/2[s] 구간에 대해 연속해서 FFT를 하여 데이터 전처리를 하였다. 따라서 데이터 수집 및 전처리 과정을 통해 지도학습 방식의 CNN 모델에서 입력하기 적합한 형태로 데이터를 생성 및 전처리하였다. 데이터 분포는 '수면 상태 정상', '약한 코골이', '심한 코골이 또는 수면 무호흡 증상' 3가지 클래스로 구분되며, 모델이 특정 클래스에 과대적합 되는것을 방지하기 위해 동일한 비율로 구성했다. 그리고 클래스 비율을 유지하면서 무작위로 추출하여 Train Data Set, Validation Data Set, Test Data Set이 약 7:2:1 비율이 되도록 분할하였다.

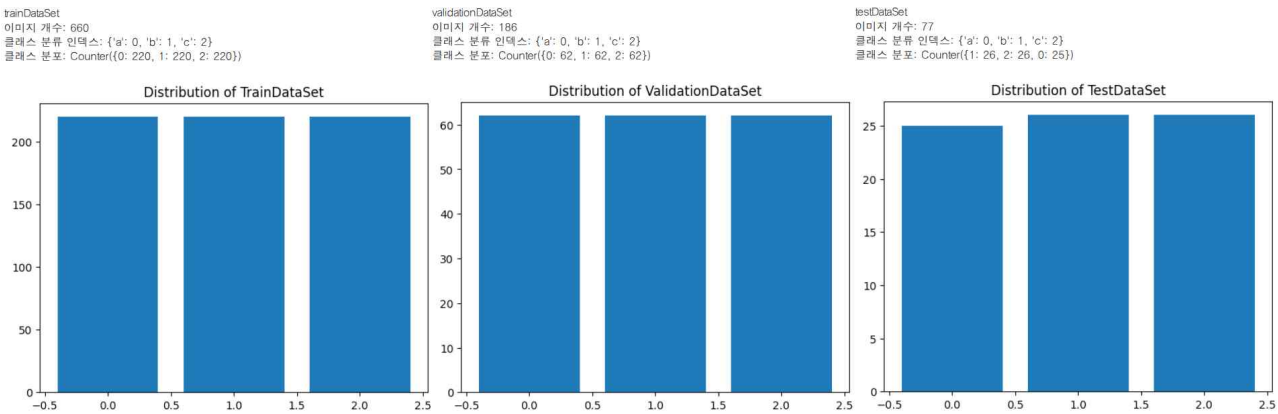

그림 2.7 데이터 분할 및 클래스별 분포(데이터 분할 비율: 7:2:1)

데이터 전처리 단계에서 가공되어진 시간, 주파수, 신호의 크기 정보가 있는 3차원 배열의 시계열 데이터를 이용해서 지도학습 방식으로 Multi Classification CNN 모델을 학습한다. 학습 데이터가 부족해서 생기는 과소적합 문제를 방지하고자 VGG16 전이학습모델의 구조와 파라미터값을 재학습하는 전이학습과 미세조정 기법으로 학습하였다. 모델의 복잡도, 분류하고자 하는 클래스 개수를 고려해서 VGG16 모델의 FC(Fully Connected Layer)를 수정하였고, 모델 앞단의 일부 Layer의 파라미터를 동결시키고 모델을 학습하였다. 과대적합 문제를 방지하고자 일부 뉴런의 가중치를 0으로 계산해서 최적화하는 Drop out 기법을 모델의 FC(Fully Connected Layer)에 적용하였고, MBGD 방법으로 Batch size를 4로 작게 설정해서 regularization과 noise를 증가시켜 모델의 복잡도를 낮추도록 최적화하였고, 단위 Epoch마다 Validation Data Set의 ACC를 계산해서 검증 정확도가 이전 Epoch보다 낮은 경우에는 파라미터를 업데이트하지 않도록 학습하였다. 그리고 Validation Data Set을 이용해서 클래스별 f1 Score가 균일한지 분석하고, 하이퍼파라미터 튜닝을 통해서 일반화 성능이 높은 모델을 선정하였다. 하이퍼파라미터는 비용함수를 CEE로 설계하고, 최적화 방식은 배치크기를 4로 하여 Adam 기법을 사용하고, 은닉층 활성화 함수는 Relu 함수를 사용해서 학습했을 때 일반화 성능이 가장 좋았다. 따라서 최종 모델의 일반화 성능은 Test Data Set에 대한 정확도를 지표로 사용하면 96%로 좋은 성능을 보였고, 해당 모델을 라즈베리파이에서 사용하기 위해 모델 구조와 파라미터를 저장하였다.

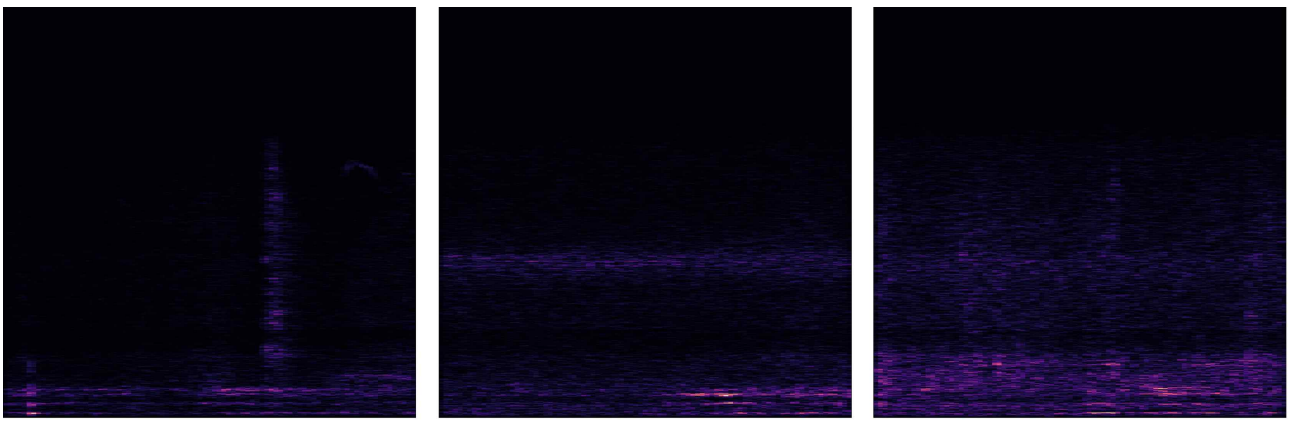

그림 2.8 STFT으로 변환된 시간, 주파수, 신호의 크기 정보가 있는 3차원 배열의 데이터
(왼쪽부터 순서대로 ‘수면 상태 정상’, ‘약한 코골이’, ‘심한 코골이 또는 수면 무호흡 증상’)

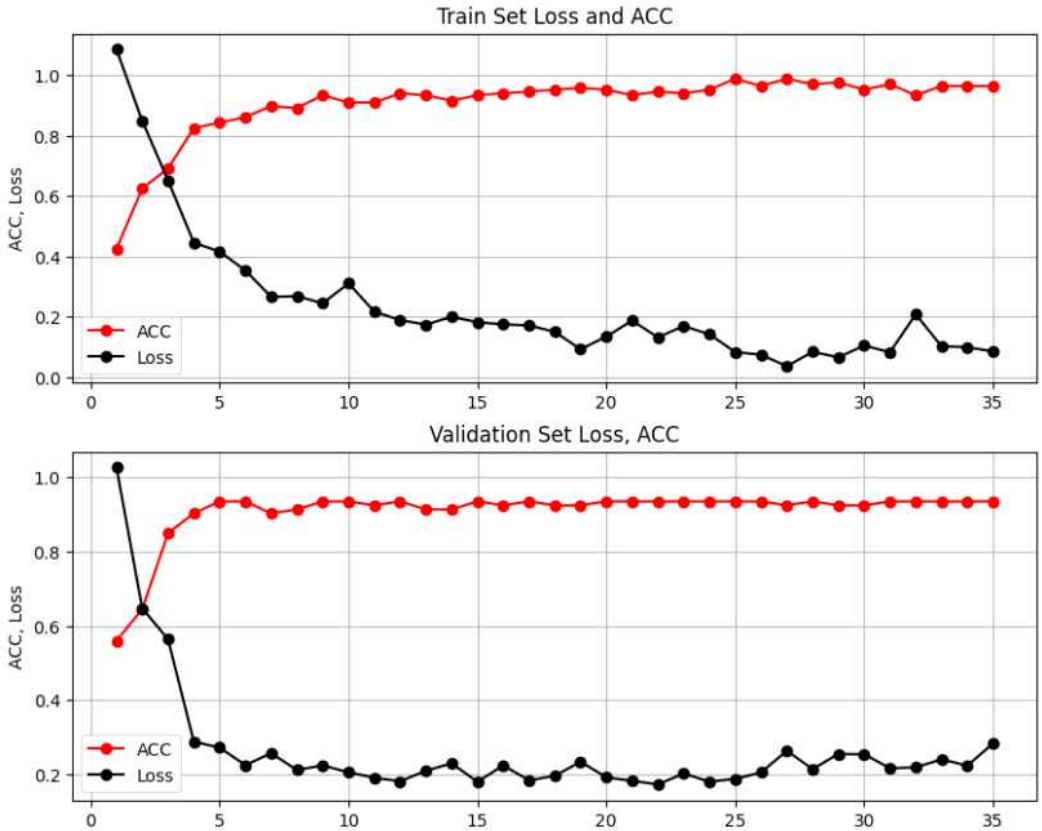

그림 2.9 과대적합 방지를 적용한 모델의 훈련 및 검증 성능

▾ 모델 일반화 성능 평가 및 모델 저장

```
def test():
    a = []
    b = []
    model_ft.eval()  # 신경망을 추론 모드로 전환
    correct = 0

    # 데이터로더에서 미니배치를 하나씩 꺼내 추론을 수행
    with torch.no_grad():  # 추론 과정에는 미분이 필요없음
        for data, targets in dataloaders["test"]:
            data = data.to(device)
            targets = targets.to(device)
            outputs = model_ft(data)  # 데이터를 입력하고 출력을 계산

            # 추론 계산
            _, predicted = torch.max(outputs.data, 1)  # 확률이 가장 높은 레이블이 무엇인지 계산
            correct += predicted.eq(targets.data.view_as(predicted)).sum()  # 정답과 일치한 경우 정답 카운트를 증가

    # 정확도 출력
    data_num = len(dataloaders["test"].dataset)  # 데이터 총 건수
    print('\n테스트 데이터에서 예측 정확도: {}/{} ({:.0f}%)\n'.format(correct,
                                                    data_num, 100. * correct / data_num))
test()
torch.save(model_ft.state_dict(), 'model222_weights.pth')
```

```
테스트 데이터에서 예측 정확도: 74/77 (96%)
```

그림 2.10 최종 모델의 일반화 성능 및 모델 저장

2.3 수면장애 진단 및 방지 시스템 설계

저전력으로 동작하는 MPU의 부족한 연산속도와 메모리 부족으로 인한 문제를 방지하기 위해 PyTorch Mobile을 이용해서 학습된 모델의 파라미터값을 양자화한다. 32bits 부동소수점 연산을 8bits 정수로 변환해서 부족한 연산속도로 인한 지연시간을 줄이고, 메모리 접근 속도를 높이는 모델 경량화 기법이다. 따라서 학습한 모델을 경량화해서 라즈베리파이에 저장하고, 라즈베리파이와 연결된 카메라와 마이크 모듈을 이용해서 이미지데이터와 음성데이터를 실시간으로 추출해서 라즈베리파이의 MPU로 데이터 전처리를 하고, 모델의 입력데이터로 사용해서 수면장애를 추론한다.

라즈베리파이에서 실시간 이미지데이터로 사람 존재 여부와 수면 자세를 판단해서 4가지 클래스에 대한 모델의 예측값을 임시 저장한다. 사람이 존재하여 수면 자세가 판단되었다면, 실시간 음성데이터를 이용해서 ‘수면 상태 양호’, ‘약한 코골이’, ‘심한 코골이 또는 수면 무호흡 증상’ 3가지 클래스에 대한 모델의 예측값을 임시 저장한다. 그리고 임시 저장된 모델별 예측값과 반복된 횟수의 조합으로 수면장애가 심한 정도를 판별하고, 상황에 적합한 제어 신호를 서보모터, 진동모터 또는 스피커에 출력해서 수면 자세를 교정하고 수면장애를 방지한다. 상황에 적합한 제어 신호는 사전에 프로그래밍된 조건 및 절차에 따라 출력되며, 모델의 예측값과 반복 횟수 조합을 정수로 인코딩한 값이 조건식에 사용된다.

진동모터와 서보모터는 사용자의 수면 질 향상에 직접적으로 사용되는 최종 제어대상이므로 카메라가 연결된 라즈베리파이와 떨어져서 위치해야 한다. 따라서 사용자의 베개에서 동작할 수 있도록 전원 공급 장치, HC-06 블루투스 모듈, 아두이노, 진동모터, 서보모터를 이용해서 회로를 구성했다. HC-06 블루투스 모듈은 무선통신 방식으로 라즈베리파이가 송신한 신호를 받아서 시리얼 통신 방식으로 아두이노에 전달한다. 제어기인 아두이노는 수신한 신호에 대응되는 제어 신호를 서보모터 또는 진동모터에 입력하는 역할을 하고, 제어대상인 모터가 동작해서 수면 자세를 교정하고 수면장애를 방지한다. 그리고 모터가 동작할 때 바운스 현상으로 인해 발생하는 문제를 방지하고자, LM7805 레귤레이터와 104 세라믹커패시터 2개를 사용하였다. 레귤레이터의 입력단자에 9[V] 전원 공급 장치와 접지되어있는 커패시터를 병렬로 연결하고, 출력 단자에 접지되어있는 커패시터를 병렬로 연결해서 안정적인 정전압을 회로에 공급하도록 하였다. 그리고 블루투스 모듈 RX pin의 전압 Logic Level은 3.3[V]에서 High이고, 아두이노 TX pin의 전압 Logic Level은 5[V]에서 High이므로 저항 1[kΩ]과 2[kΩ]을 이용해서 전압을 분배하고 연결하였다. 따라서 라즈베리파이가 실시간 데이터로 추론한 값을 이용해서 제어 신호를 결정하고 무선통신 방식으로 아두이노에 전달하면, 제어기인 아두이노가 수신한 신호에 해당하는 제어대상을 동작시켜 수면 자세와 수면장애를 개선하도록 시스템을 설계하였다.

3. 결과 및 분석

수면 자세와 수면 상태를 추론하는 모델 각각의 일반화 성능은 Test Data Set을 이용한 정확도 지표를 이용하면 94%, 96%로 높은 성능을 보였다. 학습데이터가 부족해서 생기는 과소적합 문제를 방지하기 위해 전이학습을 이용하였고, 모델의 복잡도가 높아서 학습데이터에 Overfitting 되는 문제를 방지하기 위해 학습데이터에 Random Crop을 적용하고, 모델의 FC에 Drop out 기법을 적용하고, MBGD 방식으로 Batch size를 작게 설정해서 regularization과 noise를 증가시켜 모델의 복잡도를 낮추고, 단위 Epoch 당 Validation Data Set으로 정확도를 계산해서 이전 Epoch 대비 낮은 경우에는 파라미터를 업데이트하지 않도록 학습하였다. 그리고 최적의 하이퍼파라미터를 찾기 위해 노력하였고, 일반화 성능이 높은 모델을 학습할 수 있었다.

모델의 일반화 성능을 높이기 위해, Validation Data Set으로 클래스별 모델 성능을 분석하였다. Validation Data Set에 대한 모델의 예측값과 실제값을 sklearn 라이브러리의 confusion_matrix와 classification_report를 이용해서 클래스별 모델 성능을 분석하고, 과소적합 된 클래스를 찾고, 해당 클래스의 데이터를 추가로 수집하고 재학습하여 모델 성능을 올리고자 하였다. 그러나 모델별 검증 정확도가 각각 95%, 96%로 높았고, 각각의 클래스에 대한 f1 score도 모두 0.94 이상으로 균일하게 좋은 지표를 보였기 때문에 학습 및 분석을 중단하고 최종 모델로 결정하였다. Test Data Set은 학습이 끝난 모델을 정량적으로 평가하거나 분석할 때만 이용하였다. 그림 2.11, 그림 2.12에서 볼 수 있듯이 최종 모델의 정확도는 94%, 96%이고, 클래스별 성능 지표인 정밀도, 재현율, 정밀도와 재현율의 조화 평균인 f1 Score 모두 균일하게 높은 성능을 보이는 것을 확인할 수 있다.

이미지데이터를 이용해서 학습한 최종 모델의 일반화 성능 및 분석

```
from sklearn.metrics import classification_report, ₩
confusion_matrix, precision_score, recall_score, f1_score
print(classification_report(test_y, test_y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.91	0.94	35
1	0.92	0.94	0.93	35
2	0.94	0.94	0.94	35
3	0.94	0.97	0.96	35
accuracy			0.94	140
macro avg	0.94	0.94	0.94	140
weighted avg	0.94	0.94	0.94	140

```
import seaborn as sns
import pandas as pd
cm = confusion_matrix(test_y_pred, test_y)
sns.heatmap(cm, annot = True, fmt = 'd', cmap = 'Blues')
plt.xlabel("True Label")
plt.ylabel("Predicted Label")
plt.xticks([0.5, 1.5, 2.5, 3.5], ['no people detected', 'left', 'proper', 'right'])
plt.yticks([0.5, 1.5, 2.5, 3.5], ['no people detected', 'left', 'proper', 'right'])
plt.show()
```

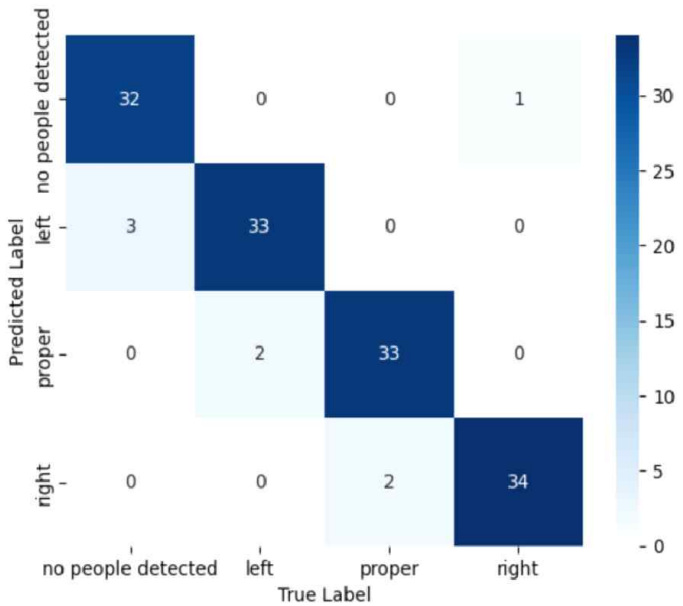

그림 2.11 이미지데이터를 이용해서 학습한 최종 모델의 일반화 성능 및 분석

음성데이터를 이용해서 학습한 최종 모델의 일반화 성능 및 분석

```
from sklearn.metrics import classification_report, ₩
confusion_matrix, precision_score, recall_score, f1_score
print(classification_report(test_y, test_y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.92	0.96	25
1	0.96	0.96	0.96	26
2	0.93	1.00	0.96	26
accuracy			0.96	77
macro avg	0.96	0.96	0.96	77
weighted avg	0.96	0.96	0.96	77

```
import seaborn as sns
import pandas as pd
cm = confusion_matrix(test_y_pred, test_y)
sns.heatmap(cm, annot = True, fmt = 'd', cmap = 'Blues')
plt.xlabel("True Label")
plt.ylabel("Predicted Label")
plt.xticks([0.5, 1.5, 2.5], ['normal sleep state', 'mild snoring', ₩
                             'severe snoring₩norsleep apnea'])
plt.yticks([0.5, 1.5, 2.5], ['normal sleep state', 'mild snoring', ₩
                             'severe snoring₩norsleep apnea'])
plt.show()
```

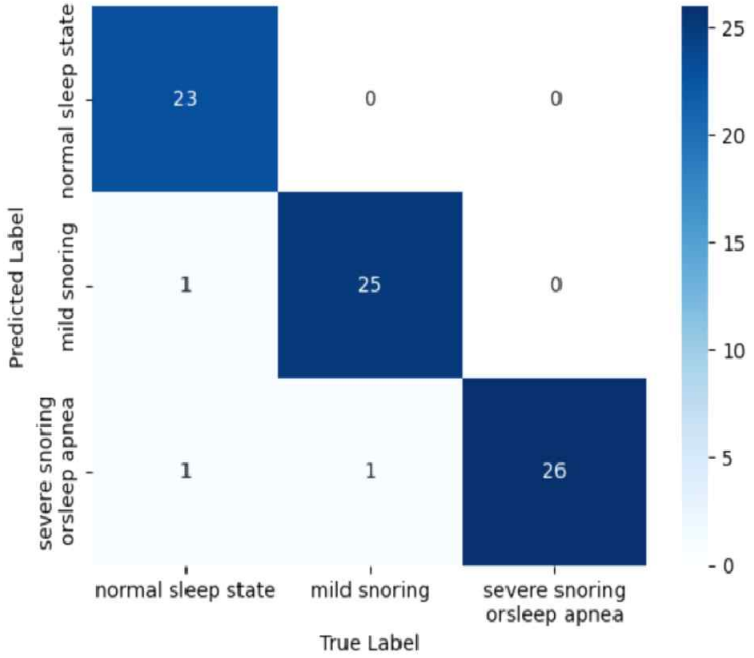

그림 2.12 음성데이터를 이용해서 학습한 최종 모델의 일반화 성능 및 분석

모델 경량화 기법을 사용해서 MCU의 부족한 연산속도로 인한 지연시간을 줄이고 메모리 접근 속도를 높였다. 실제 테스트를 했을 때 데이터를 전처리하고 추론하는데 모델마다 1~2초 정도의 시간이 소요됐다. 수면장애를 개선하는 목적을 달성하는 데는 문제가 없었으나, 실시간 처리가 중요한 상황에서는 문제가 될 수 있었다. 추후 연구에서는 지연시간을 줄이기 위해 Convolution Layer와 FC를 더 줄이거나, 모델 복잡도가 낮은 다른 알고리즘을 사용해서 모델을 설계할 필요가 있다. 그리고 모델의 파라미터값이 매우 작은 경우에는 모델 경량화 기법을 사용하였을 때 양자화 레벨의 한계로 인해 정확도가 낮아질 가능성이 있으므로 주의해야 한다.

5[V] 보조배터리를 회로의 전원 공급 장치로 사용하였을 때, 모터가 동작하면 바운스 현상으로 인해 낮은 빈도로 아두이노 또는 HC-06 블루투스 모듈이 제대로 동작하지 않는 경우가 있었다. 여러개의 배터리를 사용해서 독립적으로 연결해도 해결이 되겠지만, 회로를 소형화하고 신뢰성을 높이고자 LM7805 레귤레이터와 9[V] 건전지를 사용했다. 레귤레이터의 입력단자에 9[V] 건전지를 연결하고, 접지 되어있는 커패시터 2개를 레귤레이터의 입력단자와 출력단자에 각각 병렬로 연결해서 안정적인 5[V] 정전압을 회로에 공급하여 문제를 해결하였다.

따라서 높은 성능을 요구하는 모델 학습은 고성능 컴퓨터에서 CUDA를 이용해서 일반화 성능이 높은 모델을 효율적으로 학습하였고, 모델 경량화 기법으로 연산량과 메모리 접근 시간을 줄여서 저전력 MPU에서 실시간 데이터로 추론이 가능하게 하였다. 그리고 수면 자세를 교정하고, 수면장애를 방지하는 동작을 하는 회로를 설계하고, 여러 번 실험하며 문제점을 찾고 해결하였다. 최종 실험을 한 결과, 수면 자세와 수면장애를 정확하게 추론했고, 모델의 예측값과 반복 횟수 조합에 대응되는 동작을 제어기와 제어대상이 안정적으로 수행하는 것을 확인하였다.

4. 결론

딥러닝을 활용한 Multi Classification CNN 알고리즘으로 수면 자세와 수면장애를 추론하는 높은 성능의 모델을 학습하였다. 다양한 기법으로 과대적합 및 과소적합을 방지했고, 하이퍼파라미터 튜닝을 통하여 일반화 성능이 높은 모델을 학습하였다. Test Data Set에 대한 정확도를 지표로 사용하면 94%, 96%로 높은 성능을 보였으며, 클래스별 모델 성능은 f1 Score를 지표로 사용하면 모든 클래스에서 0.93 이상으로 균일하고 좋은 성능을 보였다.

저전력 MPU로 실시간 데이터를 추론하는 데 걸리는 시간을 최소화하였다. 모델 경량화 기법을 이용해서 데이터 타입이 float32인 최적해를 int8로 양자화하였고, 256개의 구간으로 변환된 최적해로 계산량과 지연시간을 줄였다. 그리고 양자화 오차로 인해 모델의 성능이 감소하는 것을 최소화하기 위해, regularization 효과를 높이고 모델의 복잡도를 낮추도록 학습하였다. 따라서 모델 경량화 기술을 이용해서 추론 시간은 줄이고, 모델 성능은 최대한 유지하도록 하였다.

모델이 추론한 결과를 이용해서 수면 자세를 교정하고 수면장애를 방지하는 시스템을 설계하였다. 블루투스 모듈이 무선통신 방식으로 라즈베리파이가 송신한 신호를 받으면, 시리얼 통신 방식으로 아두이노에 전달하도록 설계했다. 그리고 제어기인 아두이노는 수신한 신호에 대응되는 제어 신호를 서보모터 또는 진동모터에 입력하고, 사용자의 베개에서 제어대상이 동작하여 수면 자세를 교정하고 수면장애를 방지하도록 설계하였다.

본 연구를 수행하면서 한계점으로는 사용자 기기 내에서 AI 모델 개선이 어렵다는 단점이 있었다. 클라우드를 이용하면 쉽게 해결할 수 있지만, 클라우드 이용에 의한 통신량, 처리부하, 보안에 대한 비용이 증가한다. 따라서 문제에 적합한 해결 방법으로는 label 없이도 스스로 학습할 수 있는 자기지도학습(self supervised learning) 방식의 알고리즘을 사용하고, 특정 조건을 만족하는 경우에만 모델의 일부 파라미터값을 업데이트하도록 모델을 설계하는 것이다. 제시한 해결 방법의 장점으로는 기기 내에서 학습이 이루어지며, H/W 성능을 고려해서 최적화 빈도수를 정할 수 있고, 모델 뒷부분의 일부 layer만 최적화하도록 해서 계산량을 줄일 수 있다. 따라서 추후 연구에서는 특정 조건에서만 학습이 이루어지는 자기지도학습 방식의 모델을 설계하여, 사용자 기기 내에서 모델 개선이 쉬운 On-device AI를 개발할 필요가 있다.

참고문헌

[1] 국민건강보험 Magazine, 2020.08, https://www.nhis.or.kr/magazin/160/html/sub1.html, (accessed, 11.10.23).

[2] Chattu et al.(2018), “The Global Problem of Insufficient Sleep and Its Serious Public Health Implications”, Healthcare.

[3] 국민건강보험 Magazine, 2018.09, https://www.nhis.or.kr/magazin/137/html/c01.html, (accessed, 11.10.23).

[4] Hafner et al.(2017), “Why Sleep Matters—The Economic Costs of Insufficient Sleep”, Rand Health Quarterly.

[5] 남미래, ““밤마다 뒤척뒤척, ’꿀잠’ 잘 수만 있다면“…판 커지는 ’슬립테크’”, 머니투데이, 2023.02.14, https://news.mt.co.kr/mtview.php?no=2023021114194749942, (accessed, 11.10.23).

[6] 아주대학교산학협력단. 수면의 질 결정 장치 및 방법. 특허 출원번호 10-2010-0107335, 출원일자 2010년10월29일, 등록일자 2012년12월28일.

Abstract

AI-based Sleep Disorder Diagnosis and Prevention Technology

Tech University of Korea
Department of Electronic Engineering
Junyoung Park, Dongjoo Moon, Inho Yoo

Korea is a representative sleep-deprived country among OECD countries, and the number of patients with sleep disorders is increasing. Sleep quality is closely related to a person's physical health, mental health, life, and quality of life. People's interest in the importance of sleep and the size of the sleep tech market that seeks to improve sleep quality are rapidly increasing. And recent rapid developments in semiconductors have made AI inference possible even in low-power MPU. AI has the advantage of quickly and accurately solving complex problems based on data. And embedded systems have the advantages of portability, economic efficiency, and reliability due to their small size and weight. If we develop technology and products that improve sleep quality with On-device embedded AI that combines the strengths of AI and embedded systems, it is expected that we will be able to achieve high marketability and competitiveness in the growing sleep tech market. And it is expected that there will be many benefits not only to individuals but also to society by improving sleep disorders, which are closely related to health, life, and quality of life. Therefore, the purpose of this study is to improve sleep quality by diagnosing and preventing sleep disorders with On-device embedded AI.

The principal feature of this paper is to learn a high-performance model that infers sleeping posture and sleep disorders from image data and sound data. Additionally, model lightweighting technology that quantize the parameters of the learned model is used to reduce computational complexity and latency, and enable real-time inference on a low-power MPU. Then, the MCU uses the predicted values of the model to design a circuit that controls the speaker, servo motor, and vibration motor. Therefore, the goal is to develop technologies and systems that improve sleep quality by using real-time data to infer and prevent sleep disorders and correct sleeping posture.

Key words
Short-time Fourier transform, Deep learning, Convolutional neural network, Transfer learning,

Fine tuning, VGG16, Model compression, On-device AI