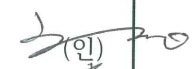



연구자(학생)	학번	2019191020	학년	본2	이름	천우영
연구과제명	(국문) CNN을 활용한 무릎 퇴행성관절염 환자의 Bone scan image 분류					
	(영문) Bone scan imaging classification of degenerative arthritis in knee joint patient using CNN					
연구 분야	<input type="checkbox"/> 기초연구분야 <input type="checkbox"/> 임상연구분야 <input checked="" type="checkbox"/> 융합연구분야 <input type="checkbox"/> 역학(코호트)연구분야 <input type="checkbox"/> 인문사회연구분야 <input type="checkbox"/> 기타 ()					
제출학기	2022년 1학기		제출일	2022년 6월 2일		
지도교수	박해정 교수님		지도교수 소속교실	핵의학교실		
연구기간	2022년 1월 22일 ~ 2022년 12월 1일 (약 11 개월)					
예상 연구비 (단위 : 천원)	인공지능 학습에 사용할 GPU 구매: 300					
<p>본인은 연구계획서 작성 과정 중 다른 사람의 아이디어, 연구과정, 결과 혹은 표현을 적절한 출처를 명기하지 않고 사용하지 않았으며, 연구멘토링 지도교수의 지도를 받아 본인 스스로 완성한 연구계획서를 제출합니다.</p> <p style="text-align: right;">제 출 자 : 천 우 영 (인) </p> <p style="text-align: right;">지도교수 : 박 해 정 (인) </p>						

요 약 문

연구과제명 (한글)		CNN을 활용한 무릎 퇴행성관절염 환자의 Bone scan image 분류		
연구과제명 (영문)		Bone scan image classification of degenerative arthritis in knee joint patient using CNN		
연구목표 (공백제외 200자 이상)		<p>환자의 병력청취나 CT, MRI로는 무릎 degenerative arthritis의 진행도를 판단하기 어렵다. 따라서 whole body bone scan image로부터 무릎 degenerative arthritis 환자의 중증도를 판단하는 convolutional neural network를 구현하여 Kellgren-Lawrence grade에 근거한 환자의 stage를 예측하고, 관절염의 진행 정도에 따른 적절한 치료계획을 수립하는데 용이하게 하고자 한다.</p>		
연구내용 (공백제외 500자 이상)		<p>1) Hyperparameter에 따른 정확도 비교 인공지능을 학습하기 위해 설정하는 초기값을 hyperparameter 라고 한다. 한 번 학습할 때 활용하는 data의 개수(Batch size), 학습 데이터 전부를 반복하는 횟수(EPOCH), 각 신경의 가중치 및 오차를 변화시키는 정도(Learning rate) 등의 hyperparameter를 변화시켜 인공지능의 성능을 높이하고자 한다.</p> <p>2) CNN layer 개수에 따른 정확도 비교 CNN에서는 이미지의 특징을 추출하기 위해 convolutional layer가 사용되는데, 이러한 layer의 개수, 즉 이미지의 특징을 얼마나 추상화하는지에 따라 정확도가 달라질 수 있다.</p> <p>3) Optimizer의 종류에 따른 정확도 비교 Optimizer란 loss function이 최소가 되는 값을 찾기 위해 사용하는 최적화 알고리즘을 의미한다. 확률적 경사 하강(Stochastic gradient descent, SGD)이 대표적인 optimizer 인데, 데이터 하나를 무작위로 추출하여 하나의 데이터에 대해 기울기를 계산하는 방식으로 loss function의 최솟값을 찾는다. SGD 이외에도 momentum, Adagrad, Adam 등의 optimizer를 사용해보면서 loss function이 최소가 되는 최적화 알고리즘을 찾는다.</p>		
연구개발에 따른 기대성과 (공백제외 200자 이상)		<p>염증에 대한 민감도가 높은 bone scan을 가지고 degenerative arthritis의 진행도를 예측하면 선행연구보다 더 좋은 정확도를 내는 인공지능을 만들 수 있을 것이다. 또한 CNN은 bone scan 이미지에서 특징을 추출하여 학습하는 algorithm이며, 해상도가 낮아 구조를 파악하기 어려운 bone scan image를 분류할 수 있다는 점에서 MRI로 stage를 진단하는 의사보다 더 탁월한 진단 능력을 가질 것으로 예상된다.</p>		
색인어 (key word)	국문	합성곱 신경망	퇴행성관절염	뼈 스캔
	영문	Convolutional neural network	Degenerative arthritis	Bone scan

1. 연구 배경

퇴행성관절염(Osteoarthritis, OA 또는 degenerative arthritis)은 뼈의 과증식, 윤활액 염증, 연골하 경화증, 관절 사이 좁아짐 등을 동반하는, 관절연골에서 발생하는 모든 퇴행성 질환을 일컫는다.[1] 퇴행성관절염은 전 세계 약 3억 명이 앓고 있는 흔한 노인성 질병 중 하나이며, 기대수명이 증가함에 따라 유병률과 발생률이 크게 늘고 있다.[2,3] 퇴행성관절염의 진단은 병력 청취와 X-ray 검진을 통해 이루어지지만, 이러한 방식은 초기 관절염을 진단하기 어렵다는 단점이 있다. 뼈와 같이 큰 조직의 구조만을 감별할 수 있는 X-ray의 특성상 cartilage나 meniscus와 같은 연부조직의 구조를 파악하는 데는 기술적 한계가 있다.

그러나 핵의학 영상은 이러한 문제점을 극복할 수 있다. PET이나 Bone scan과 같은 핵의학 영상은 병변의 생리학적 변화로 인한 방사성 표지자의 흡수 변화를 통해 병변의 위치를 알아낸다. 이는 다른 영상의학 기기보다 민감도가 훨씬 뛰어나기 때문에, 초기 퇴행성관절염 여부도 쉽게 판단할 수 있다. 그래서 초기 퇴행성관절염 여부를 판단하기 위해 핵의학 영상을 사용해보아야겠다는 생각이 들었다.

또한, 최근 인공지능 기술이 각광을 받으면서 의학적 진단을 내리는 인공지능에 대한 연구도 활발히 진행되고 있다. CNN(Convolutional neural network, 합성곱 신경망)은 인공지능 분야에서 가장 잘 알려진 신경망으로, MRI와 같은 이미지를 분류하는 인공지능을 구현할 때 사용한다.[4] 기존의 신경망 구조는 m 개의 입력과 n 개의 출력이 모두 연결되어 $m \times n$ 개의 신경으로 구성된 fully connected layer의 형태인데, 이는 계산이 매우 복잡할뿐더러 이미지 분류에 불필요한 정보들까지 계산해야한다는 단점이 있다. CNN은 이를 보완하기 위해 여러 차례의 이미지 특징 추출 과정을 거쳐 데이터를 단순화한 후, fully connected layer를 통과시키면 이미지의 거시적인 특징만을 가지고 학습할 수 있게 된다.

이러한 CNN 구조를 바탕으로 bone scan 이미지를 학습하는 인공지능을 구현하여 학습시키면 퇴행성관절염 환자의 이미지에서 특징적인 부분만 추출하여 학습하므로, 기존의 진단보다 더 정확하게 관절염 진행도를 예상할 수 있을 것이다. 인공지능을 100% 믿어서는 안 되겠지만, 모든 인공지능이 그렇듯 진단을 내릴 때 좋은 참고자료로 활용할 수 있을 것으로 기대된다. 이에 CNN을 기반으로 한 무릎 퇴행성관절염 환자의 bone scan 이미지 분류를 주제로 연구를 계획하게 되었다.

2. 연구 필요성

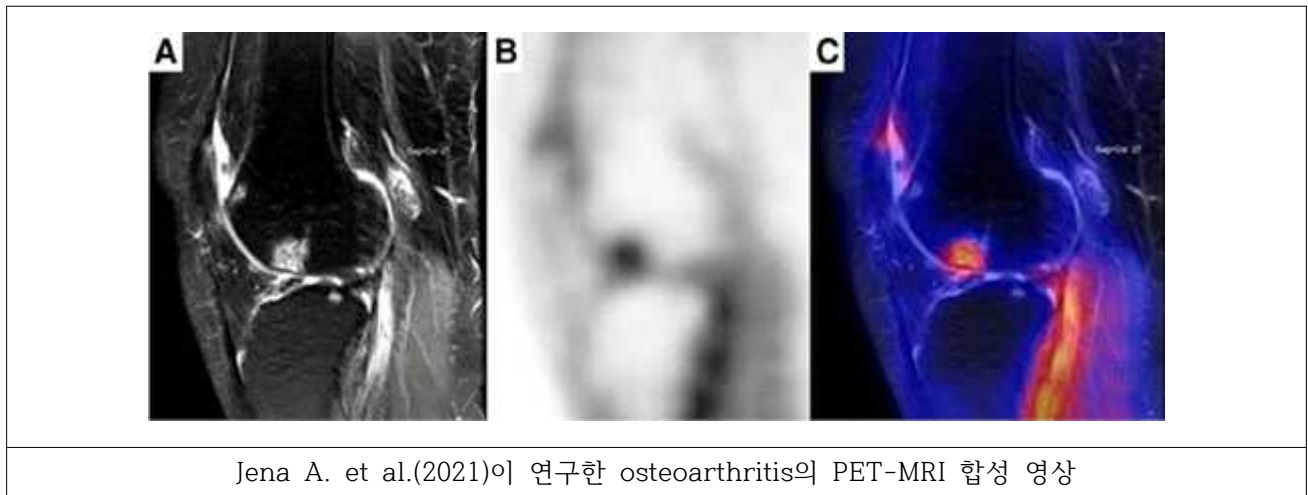
1) 알려진 내용

Petersson IF et al.(1998)은 4년 이상 무릎 관절염으로 통증을 겪은 환자들을 대상으로 혈청 cartilage oligomeric matrix protein(COMP)과 bone sialoprotein(BSP) 농도를 조사하였고, 그 결과 이들이 퇴행성관절염의 biomarker로 사용할 수 있음을 밝혀냈다.[5] 이들 biomarker는 관절염이 발생했을 때 농도가 증가하기 때문에 초기 퇴행성관절염을 진단하는데 유용하게 사용될 수 있다.

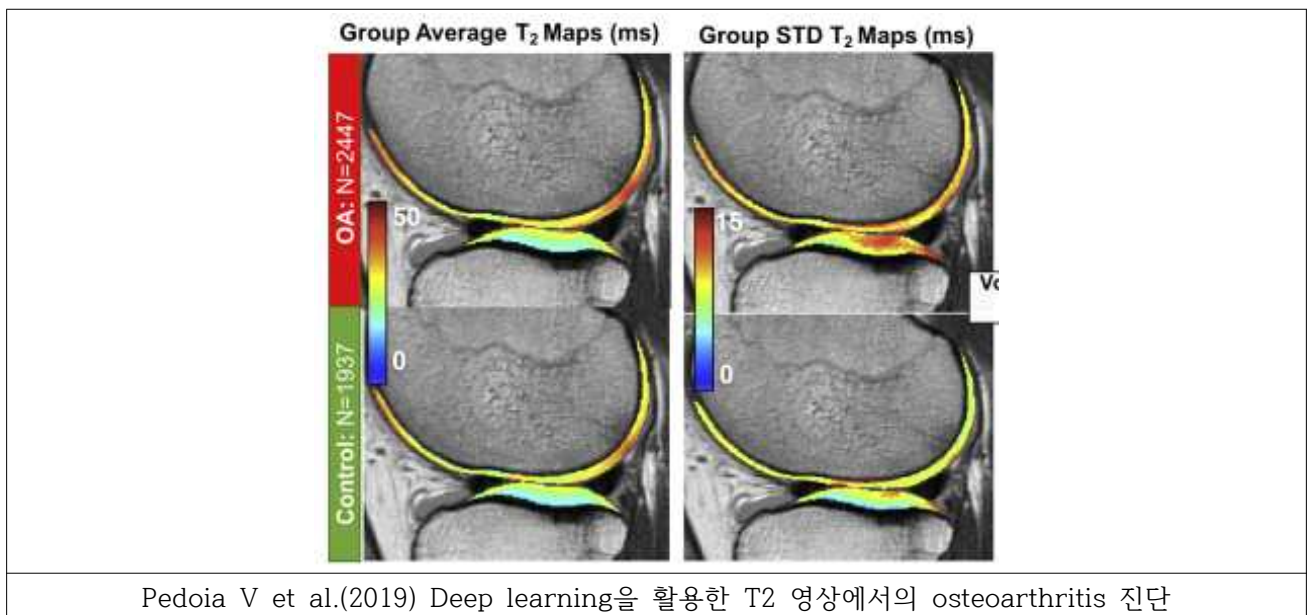
퇴행성관절염은 임상적으로 진단을 내리는 것이 보통이나, 퇴행성관절염을 조기에 진단하거나 병리 기전을 규명하려는 경우 PET이나 MRI 촬영이 필요하다.[6] 특히, 퇴행성관절염 초기에 일어나는 연부 조직의 미세한 변화는 X-ray로는 잘 나타나지 않기 때문에, PET이나 MRI와 같은 정밀한 장비들을 사용해야 관찰할 수 있다.

Jena A et al.(2021)가 진행한 연구에서는 PET과 MRI 영상을 합성하여 무릎의 퇴행성관절염이 일어난 정확한 위치를 알아내는 연구를 진행하였다. 해상도가 높은 MRI를 통해 무릎의 해부학적 정보를 얻고, 민감도가 높은 PET을 통해 무릎의 병변 위치를 얻어 두 이미지를 합치는 방식을 택

했다. 그 결과, 관절 구조에서의 혈액 공급과 대사 변화를 민감하게 관찰하여 무릎 관절의 퇴행성 변화를 감지할 수 있었다.[1]



한편, deep learning을 활용해 퇴행성관절염을 진단하는 선행 연구도 있다. Norman B et al.(2018), Tack A et al.(2018)은 무릎 MRI를 바탕으로 연골과 뼈, 반달 연골(meniscus)에 대한 segmentation을 진행하였다[7,8]. 또한, Pedoia V et al.(2019)이 진행한 연구에서는 voxel 기반의 relaxometry 분석을 진행하기 위해, deep learning 모델과 기존의 기계학습 모델의 퇴행성관절염 환자과 대조군 구분 정확도를 비교하였다. 그 결과 deep learning 모델의 AUC는 0.83, 기존의 기계학습 모델의 AUC는 0.78로 deep learning model이 더 정확했다.[9]



2) 알려지지 않은 내용

Jena A. et al.(2021)의 연구에서는 PET과 MRI의 서로 다른 두 이미지를 단순 합성했을 뿐, CNN이나 deep learning 등의 인공지능을 활용하지는 않았다. 이런 경우 단순 합성 이미지를 바탕으로 의사가 환자의 병기를 판단해야 하는데, 2차원 사진만을 가지고는 퇴행성관절염의 정확한 병기를 진단하기 어렵다. 따라서 인공지능이 이미지를 분석한 후 예상되는 병기를 도출하여 의사의 진단을 보조해야할 것이다.

Pedoia V et al.(2019)이 연구에서 사용한 deep learning을 활용한 MRI image segmentation은

해상도는 좋으나 cartilage와 meniscus의 두께만으로 관절염의 경증 여부를 판단한다. Cartilage와 meniscus의 두께는 개인차가 존재하기 때문에, 얇은 meniscus를 가진 정상인 사람을 인공지능은 비정상으로 판단할 수 있다. 따라서 무릎 관절 구조의 두께보다는 민감도가 높은 핵의학 영상을 통한 염증 detection 연구가 필요하다. 또한, 이 연구에서는 이미지 분류에 널리 사용되는 CNN을 활용하지 않았다.

3) 필요성

퇴행성관절염은 유병률이 매우 높은 질환임에도 불구하고, CT나 MRI 등의 기존 진단법으로 초기 관절염을 진단하는 데는 한계가 있다. 따라서 염증에 대한 민감도가 높은 bone scan을 가지고 퇴행성관절염의 진행도를 예측하는 연구가 필요하다. 또한 CNN은 해상도가 낮아 의사가 알아보기 어려운 bone scan 이미지를 분석한 후, 환자에게서 예상되는 병기를 제시할 수 있다는 점에서 매력적인 인공지능이다.

3. 연구 목적

Whole body bone scan image로부터 무릎 퇴행성관절염 환자의 중증도를 판단하는 convolutional neural network를 구현하여 환자의 병기를 예측하고, 관절염의 진행 정도에 따른 적절한 치료계획을 수립하는데 용이하게 하고자 한다.

4. 연구 내용

1) Hyperparameter에 따른 정확도 비교

인공지능을 학습하기 위해 설정하는 초기값을 hyperparameter 라고 한다. Hyperparameter를 얼마로 설정하느냐에 따라 모델의 학습 속도와 정확도가 크게 달라지기 때문에 가장 적절한 hyperparameter를 찾는 것을 목표로 한다. Hyperparameter의 예시는 다음과 같다.

- Batch size: 한 번 학습할 때 활용하는 data의 개수
- Epoch: 학습 데이터 전부를 반복하는 횟수
- Learning rate: 각 신경의 가중치 및 오차를 변화시키는 정도

2) CNN parameter에 따른 정확도 비교

CNN에서는 이미지의 특징을 추출하기 위해 convolutional layer가 사용되는데, 이미지의 특징을 어떻게 추상화하는지에 따라 인공지능의 성능이 달라질 수 있다.

- Channel: 입력, 출력 값의 개수
- Filter(Kernel): 합성곱 계산 시 적용할 parameter. Filter size가 3이면 3*3 필터를 사용한다.
- Stride: 합성곱 계산 시 필터가 이동하는 pixel 수
- Padding: 이미지 데이터 바깥에 특정 값을 추가해서 계산할 때 추가할 pixel의 층 수
- Pooling: 합성곱 계산이 완료된 데이터에서 대푯값을 추출하는 과정.

3) Optimizer의 종류에 따른 정확도 비교

Optimizer란 loss function이 최소가 되는 값을 찾기 위해 사용하는 최적화 알고리즘을 의미한다. 확률적 경사 하강(Stochastic gradient descent, SGD)이 대표적인 optimizer 인데, 데이터 하나를 무작위로 추출하여 하나의 데이터에 대해 기울기를 계산하는 방식으로 loss function의 최소값을 찾는다. 만약 Loss가 최소가 아닌 극소점을 향해 학습이 진행된다면 인공지능의 정확도가 떨어진다.

어지게 된다. 이러한 문제를 해결하기 위해 step size와 step 방향을 다양하게 변화시킨 optimizer 들이 많이 개발되었다. SGD 이외에도 momentum, Adagrad, Adam 등의 optimizer를 적용해보 면서 가장 적절한 optimizer를 찾고자 한다.

5. 연구 방법과 추진 계획

1) Google colab

Google colab(colab)은 Google이 제공하는 무료 Jupyter notebook 환경을 말한다. Jupyter notebook은 인터넷상에서 오픈소스 소프트웨어를 제공하는 프로젝트를 의미하는데, python을 컴퓨터에 설치하지 않아도 바로 코드를 실행할 수 있다는 장점이 있다. Amazon사의 SageMaker notebooks, Microsoft사의 Azure notebook도 Jupyter notebook을 제공하고 있 으나, 라이브러리가 많고 Google drive에 있는 파일과 연동해서 사용할 수 있다는 장점 때문에 Google colab을 많이 사용한다.

Image data mounting

```
[1] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Load dataset

```
[2] from scipy import io
import os
import numpy as np
import matplotlib.pyplot as plt

matlist = os.listdir('/content/drive/My Drive/11')
matlist.sort()
```

Google drive에 있는 데이터를 연동하여 학습을 진행시킬 수 있다.

2) Pytorch

Pytorch는 Facebook에서 개발한 Torch 라이브러리 기반의 python 오픈소스 기계학습 framework이다.[11] Dataset, dataloader, 모델 구성, data transform, 신경망 구성에 이르기까 지 인공지능을 구현하기 위해 필요한 명령어들을 담고 있다. Tensorflow, Keras와 더불어 인공지 능 분야에서 가장 많이 사용되고 있는 라이브러리 중 하나인데, 최근 Pytorch로 신경망을 개발하 는 연구가 늘고 있는 추세로 보아 Pytorch가 연구에 적합한 라이브러리라고 생각했다.

Import torch

```
[93] import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

import torchvision
from torchvision import datasets, models, transforms
from torch.utils.data import Dataset, DataLoader
import time
```

Setup device

```
[94] if torch.cuda.is_available():
    device = torch.device('cuda')
else:
    device = torch.device('cpu')

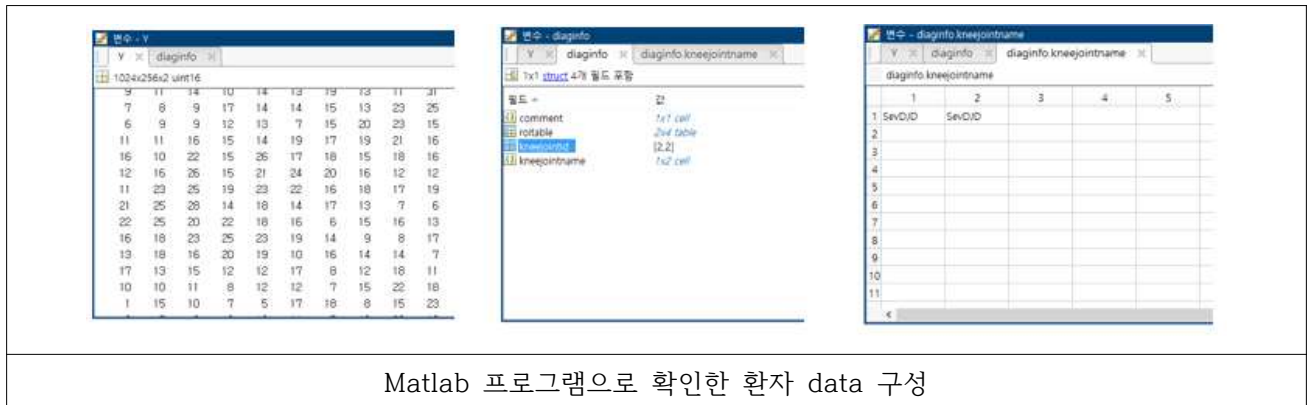
print('device:', device)

device: cpu
```

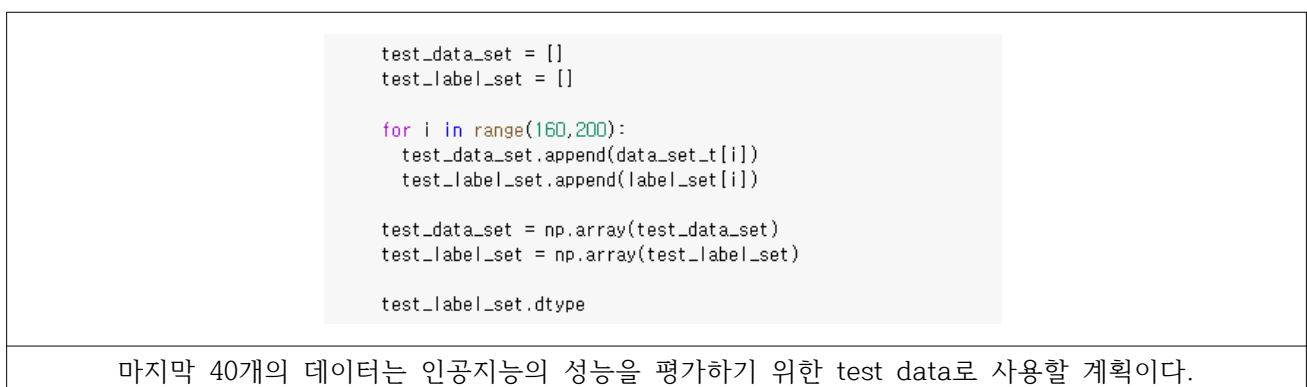
위는 Pytorch 라이브러리를 설치하는 소스 코드이다.
아래는 인공지능을 학습할 장치를 확인하는 소스 코드이다.

3) 연구 자료 확보 및 활용 계획

무릎의 bone scan data는 강남세브란스 병원의 전태주 핵의학과 교수님, 용인세브란스 병원의 정용휴 핵의학과 교수님으로부터 205명의 환자 data를 제공받을 예정이다. 각 환자의 data는 크게 Y, info, diaginfo로 구성되어 있다. Y에는 환자의 whole body bone scan data가 1024*256 크기의 uint16(65535 이하의 음이 아닌 정수) 형태로 저장되어 있고, info에는 file format, patient ID, data size, color type, 주입한 방사성 동위원소 등의 정보가 저장되어 있다. Diaginfo에는 무릎 관절에 발생한 퇴행성관절염의 중증도가 6단계(Normal, minimal, mildDJD, modDJD, sevDJD, unspect)로 기술되어 있고, 그 단계는 각각 2~6 사이의 정수 및 -1로 표기되어 있다. 임상 진단은 퇴행성관절염의 표준 분류 기준인 Kellgren-Lawrence grade의 기준을 따랐다.[10]



실제 연구에서 사용할 data set은 whole body bone scan data 중 양 무릎을 포함한 128*256 size의 image data이며, 학습을 시킬 때는 이를 양쪽으로 나누어 왼쪽 무릎과 오른쪽 무릎(각각 128*128 size)을 개별적으로 학습시킬 예정이다. 또한, 마지막 40개의 data는 test data, 나머지 data는 학습 데이터로 구성하였다.



4) Hyperparameter

Batch size는 통상적으로 2의 거듭제곱으로 나타낼 수 있는 정수로 설정하므로, 32를 연구의 표준으로 둘 것이다. Epoch은 학습 데이터가 적어 100을 기준으로 두었다. Learning rate은 0.001로 설정하였다.

CNN layer 개수 기준은 3개이고, CNN layer 개수에 따른 정확도를 비교할 때는 1개, 2개, 3개로 바꾸어서 정확도를 측정하려고 한다.

Optimizer의 기준은 stochastic gradient descent(SGD)로, momentum, Adagrad, Adam 등의 optimizer를 적용해보고자 한다.

각 실험에서 통제변인은 상기한 기준 값들이며, 조작변인은 기준 값들에서 2배, 3배로 변화해가면서 설정할 것이다.

```
[96] learning_rate = 0.001
      batch_size = 32
      num_epochs = 100
```

이 연구에서 성능 기준으로 사용할 hyperparameter 값

5) CNN 구조

CNN은 convolutional layer와 fully connected layer로 구성된다. 모든 convolutional layer의 filter(kernel) size는 3*3이며, padding은 1로 설정하였다. 각 층은 합성곱 연산, 배치 정규화, 활성화 함수(ReLU), 최대값 추출(Maximum pooling)의 과정을 거친다. 2*2 단위로 pooling을 거치기 때문에 layer를 한 번 거칠 때마다 pixel size가 2배씩 감소한다.

- 첫 번째 합성곱 층: 하나의 입력 값을 넣어 16개의 서로 다른 결과를 출력한다.
- 두 번째 합성곱 층: 16개의 입력 값을 넣어 32개의 서로 다른 결과를 출력한다.
- 세 번째 합성곱 층: 32개의 입력 값을 넣어 64개의 서로 다른 결과를 출력한다.
- 완전 연결 층: 64개의 16*16 data가 입력되어 5개(grade)의 서로 다른 결과를 출력한다.

```
[97] class SimpleCNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv_layer1 = nn.Sequential(
            nn.Conv2d( in_channels=1, out_channels=16, kernel_size=3, padding=1 ),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.conv_layer2 = nn.Sequential(
            nn.Conv2d( in_channels=16, out_channels=32, kernel_size=3, padding=1 ),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.conv_layer3 = nn.Sequential(
            nn.Conv2d( in_channels=32, out_channels=64, kernel_size=3, padding=1 ),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.fc_layers = nn.Sequential(
            nn.Linear(16*16*64, 5),
            nn.ReLU(),
        )

    def forward(self, x):
        x = self.conv_layer1(x)
        x = self.conv_layer2(x)
        x = self.conv_layer3(x)

        x = x.view( x.size(0), -1 ) # flatten
        x = self.fc_layers(x)
        return x
```

연구에 사용할 CNN의 구조.

3개의 convolutional layer와 1개의 fully connected layer로 구성되어 있다.

6. 결과분석 방법(통계분석)

1) Cross entropy loss

인공지능을 학습시킬 때는 우리가 원하는 답이 도출되는 방향으로 학습시켜야 한다. CNN과 같은 지도학습은 실제 답을 주고, 인공지능이 도출한 답과 다르면 각 신경망에 할당된 가중치를 조금씩 바꾸어 나가면서 인공지능을 학습시킨다. 이 때 실제 답과 인공지능이 도출한 답이 얼마나 다른지를 수치화 한 것이 '손실 함수(Loss function)'이다. 손실 함수에는 회귀분석에 사용하는 평균제곱 오차(mean squared error, MSE)와 cross entropy loss가 있는데, CNN과 같은 분류 문제는 주로 cross entropy loss를 사용한다. 결과적으로 CNN은 cross entropy loss를 최소화하는 방향으로 학습을 진행하게 된다.

인공지능이 학습할 때 환자의 실제 grade(정답)는 one-hot-encode 방식으로 제공된다. One-hot-encode란, 정답에 해당하는 label을 1로, 오답에 해당하는 나머지 label을 0으로 저장하는 방식을 의미한다. 예를 들어, moderate degenerative arthritis 환자를 4번째 label로 분류했다면, 이 환자를 학습할 때 정답은 [0, 0, 0, 1, 0]으로 표현된다.

Cross entropy loss의 식은 다음과 같다.

$$L = -\frac{1}{N} \sum_n \sum_k t_{nk} \log y_{nk}$$

t_{nk} 는 one-hot encode 방식으로 저장된 n 번째 data의 k 번째 답을 의미하여, 정답에 해당하는 grade는 1, 오답에 해당하는 grade는 0의 값을 가지게 된다. y_{nk} 는 인공지능이 예상한 결과 값으로, 환자 data가 각 grade에 해당될 확률을 표현한 것이다.[12] 인공지능이 정답에 가까운 답을 도출할수록 손실 함수의 값은 작아지므로 손실 함수의 값이 최소가 되는 지점을 향하도록 학습이 이루어진다.

Model 초기화

```
[98] model = SimpleCNN()
    model.to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

[99] from torch.utils.data import DataLoader

    train_dataloader = DataLoader(data_set_t, batch_size=batch_size, shuffle=True)
    test_dataloader = DataLoader(test_data_set, batch_size=batch_size, shuffle=True)

    # dataiter = iter(train_dataloader)
    # images, labels = dataiter.next()
```

Model과 dataloader를 초기화하고, 학습할 때 사용할 loss function과 optimizer를 설정한다.

2) Accuracy(정확도)

데이터를 학습하는 과정에서 1 epoch이 끝날 때마다 cross entropy loss를 계산하고, 앞서 설정한 20개의 test data를 가지고 모델의 정확도를 평가하고자 한다. 정확도는 20개의 test data 중 몇 명의 환자를 예측하는데 성공하는지 정답률로 표현될 것이다.

```
[101] model.train()
start_time = time.time()

for epoch in range(num_epochs):
    running_loss = 0
    running_corrects = 0

    # 학습 데이터 불러오기
    for inputs, labels in train_dataloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        # 모델에 입력 후 결과 계산
        optimizer.zero_grad()
        outputs = model(inputs.float())
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

        # 기울기 계산 및 학습
        loss.backward()
        optimizer.step()

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

    epoch_loss = running_loss / len(data_set_t)
    epoch_acc = running_corrects / len(data_set_t) * 100.
```

```
#1 Loss: 1.6243 Acc: 18.9055% Time: 5.4364s
#2 Loss: 1.6026 Acc: 22.8856% Time: 10.8021s
#3 Loss: 1.5917 Acc: 23.1343% Time: 16.1856s
#4 Loss: 1.5786 Acc: 23.1343% Time: 21.5263s
#5 Loss: 1.5630 Acc: 23.3831% Time: 26.9068s
#6 Loss: 1.5580 Acc: 23.6318% Time: 32.2438s
#7 Loss: 1.5540 Acc: 23.3831% Time: 37.6338s
#8 Loss: 1.5557 Acc: 23.8806% Time: 42.9706s
#9 Loss: 1.5393 Acc: 23.8806% Time: 48.3286s
#10 Loss: 1.5460 Acc: 24.1294% Time: 53.6895s
#11 Loss: 1.5173 Acc: 25.6219% Time: 59.0446s
#12 Loss: 1.5182 Acc: 23.8806% Time: 64.3798s
#13 Loss: 1.4916 Acc: 27.1144% Time: 69.7377s
#14 Loss: 1.4957 Acc: 27.6119% Time: 75.0919s
#15 Loss: 1.4827 Acc: 26.8657% Time: 80.4539s
#16 Loss: 1.4760 Acc: 26.1194% Time: 85.8066s
#17 Loss: 1.4536 Acc: 27.3632% Time: 91.1647s
#18 Loss: 1.4447 Acc: 28.3582% Time: 96.4952s
#19 Loss: 1.4421 Acc: 30.0995% Time: 101.8619s
#20 Loss: 1.4307 Acc: 28.3582% Time: 107.2220s
```

(좌) 1 epoch마다 batch size만큼의 데이터를 선택하여 model을 학습하고, loss와 정확도를 계산한다.
(우) 기존 hyperparameter로 학습을 진행시킨 모습. 아직 정확도가 높지 않은 것을 확인할 수 있다.

7. 연구 진행 체계 및 일정

2022.01.22	첫 Lab meeting (교수님 연구실 소개)
2022.02.17	두 번째 Lab meeting
2022.03.28	세 번째 Lab meeting (주제 선정)
2022.04.28	Bone scan image 전달
2022.05.17	중간 발표
2022.06.02	연구 계획서 제출
2022.07~9	설정된 변인에 따라 인공지능 학습 진행
2022.10	결과 정리 및 해석
2022.11~12	보고서 작성

8. 참고문헌

- [1] Jena A, Taneja S, Rana P, Goyal N, Vaish A, Botchu R, Vaishya R. Emerging role of integrated PET-MRI in osteoarthritis. *Skeletal Radiol*. 2021 Dec;50(12):2349-2363.
- [2] Abramoff B, Caldera FE. Osteoarthritis: Pathology, Diagnosis, and Treatment Options. *Med Clin North Am*. 2020 Mar;104(2):293-311.
- [3] Pereira D, Ramos E, Branco J. Osteoarthritis. *Acta Med Port*. 2015 Jan-Feb;28(1):99-106.
- [4] Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". *Mathematics and Computers in Simulation*. Elsevier BV. 177: 232-243.
- [5] Petersson IF, Boegård T, Dahlström J, Svensson B, Heinegård D, Saxne T. Bone scan and serum markers of bone and cartilage in patients with knee pain and osteoarthritis. *Osteoarthritis Cartilage*. 1998 Jan;6(1):33-9.
- [6] Goodman S. Osteoarthritis. In: Yee AMF, Paget SA, editors. *Expert Guide to Rheumatology Arthur Yee Expert Guide to Rheumatology*, Stephen Paget American College of Physicians. Nurs Older People. 2005 Dec 1;269-283.
- [7] Tack A, Mukhopadhyay A, Zachow S. Knee menisci segmentation using convolutional neural networks: data from the Osteoarthritis Initiative. *Osteoarthritis Cartilage*. 2018 May;26(5):680-688.
- [8] Norman B, Pedoia V, Majumdar S. Use of 2D U-Net Convolutional Neural Networks for Automated Cartilage and Meniscus Segmentation of Knee MR Imaging Data to Determine Relaxometry and Morphometry. *Radiology*. 2018 Jul;288(1):177-185.
- [9] Pedoia V, Lee J, Norman B, Link TM, Majumdar S. Diagnosing osteoarthritis from T2 maps using deep learning: an analysis of the entire Osteoarthritis Initiative baseline cohort. *Osteoarthritis Cartilage*. 2019 Jul;27(7):1002-1010.
- [10] Misir A, Yildiz KI, Kizkapan TB, Incesoy MA. Kellgren-Lawrence grade of osteoarthritis is associated with change in certain morphological parameters. *Knee*. 2020 Jun;27(3):633-641.
- [11] Ketkar, Nikhil. *Introduction to PyTorch*". *Deep Learning with Python*. Berkeley, CA: Apress; 2017. p.195-208.
- [12] 안세진. Modified cross-entropy loss function in neural network. [master's thesis]. [Seoul]: Graduate School, Yonsei University; 2018. Korean.