

CNN을 활용한 퇴행성 관절염 환자의 Bone scan image 분류

본과 2학년 천우영

목 차

- 연구 배경
- 연구 목적
- 연구 내용
- 연구 방법 및 일정
- 기대 효과

연구 배경

Degenerative joint disease(DJD)

- 관절을 보호하고 있는 연골의 손상 혹은 퇴행성 변화로 관절을 이루는 뼈와 인대 등에 손상이 일어나고 염증과 통증이 생기는 질환
- 증상: 관절의 통증 호소, 조조강직, 관절의 운동제한
- 진단: 증상과 진찰 소견, X선 검사
- Osteoarthritis: 골관절염 / Degenerative arthritis: 퇴행성 관절염

연구 배경

Limitation of X-ray diagnosis

- 초기 관절염의 진단에 한계
- 관절 연부조직의 변화나 부종 진단에 한계
- 초음파, MRI 검사, **Bone scan, PET** 이 대안이 될 수 있음

연구 배경

Convolutional neural network (CNN)

- 합성곱 신경망
- 이미지 분류에 가장 많이 사용되는 신경망
- 이미지의 특징을 추출(pooling)하여 계산 시간이 적음

→ CNN을 활용한 초기 퇴행성관절염 진단!

연구 목적

이전 연구

Emerging role of integrated PET-MRI in osteoarthritis

Amarnath Jena¹ · Sangeeta Taneja¹ · Prerana Rana^{1,2} · Nidhi Goyal³ · Abhishek Vaish⁴ · Rajesh Botchu⁵ ·
Raju Vaishya⁴

Received: 6 April 2021 / Revised: 16 June 2021 / Accepted: 16 June 2021 / Published online: 29 June 2021
© ISS 2021

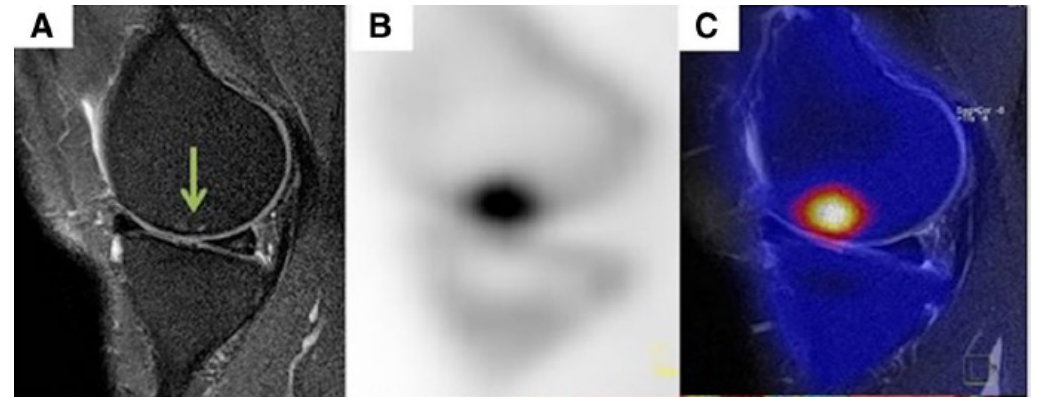
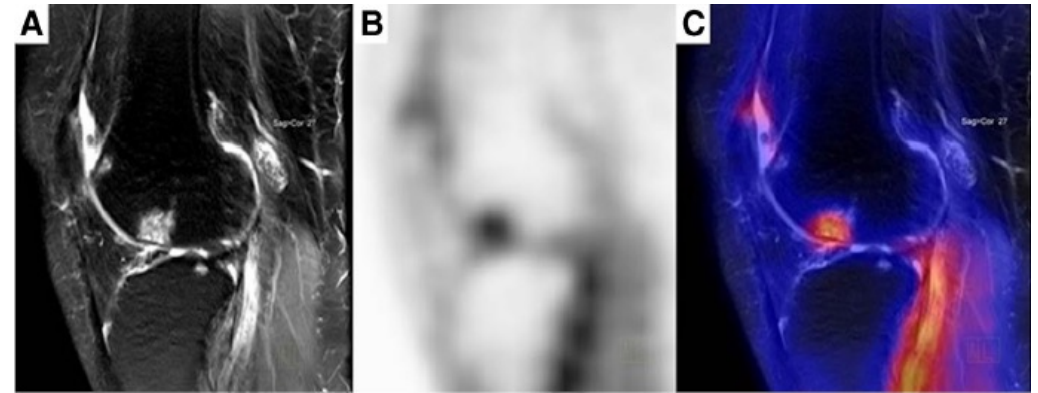
Abstract

Osteoarthritis (OA) is a common degenerative disorder of the articular cartilage, which is associated with hypertrophic changes in the bone, synovial inflammation, subchondral sclerosis, and joint space narrowing (JSN). Radiography remains the first line of imaging till now. Due to the lack of soft-tissue depiction in radiography, researchers are exploring various imaging techniques to detect OA at an early stage and understand its pathophysiology to restrict its progression and discover disease-modifying agents in OA. As the OA relates to the degradation of articular cartilage and remodeling of the underlying bone, an optimal imaging tool must be sensitive to the bone and soft tissue health. In that line, many non-invasive imaging and minimally invasive techniques have been explored. Out of these, the non-invasive compositional magnetic resonance imaging (MRI) for evaluation of the integrity of articular cartilage and positron emission tomography (PET) scan with fluoro-deoxyglucose (FDG) and more specific bone-seeking tracer like sodium fluoride (¹⁸F-NaF) for bone cartilage interface are some of the leading areas of ongoing work. Integrated PET-MRI system, a new hybrid modality that combines the virtues

연구 목적

이전 연구

PET-MRI is emerging as a useful tool that can be used in the research setup. Several studies support its efficacy. It proved to be highly sensitive to perfusion and metabolic alterations in joint structures, detect degenerative changes in cartilage associated with increased turnover in the adjacent bone simultaneously, and metabolic anomalies in the subchondral bone that appear normal on MRI. Two radi-



연구 목적

이전 연구

Osteoarthritis and Cartilage

OAR

Review

Osteoarthritis year in review 2019: imaging

R. Kijowski ^{†*}, S. Demehri [‡], F. Roemer [§] ||, A. Guermazi [§]

[†] Department of Radiology, University of Wisconsin–Madison, Madison, WI, USA

[‡] Department of Radiology, Johns Hopkins University, Baltimore, MD, USA

[§] Department of Radiology, Boston University, Boston, MA, USA

|| Department of Radiology, Friedrich-Alexander University Erlangen-Nürnberg (FAU), Universitätsklinikum Erlangen, Erlangen, Germany

Deep learning in OA imaging

Deep learning is one of the most exciting new fields in medical imaging⁸⁷. Unlike traditional machine learning, deep learning does not require the extraction of predefined information from imaging studies. Given enough training data, deep learning can automatically learn a representative subset of features that may not even be perceivable by the human eye. Deep learning methods have been recently used for rapid fully-automated segmentation of cartilage and bone^{88–90}, meniscus^{89,91}, and all joint structures (Fig. 5)⁸⁸ on knee MRI. Deep learning techniques have been described for identifying structural pathology within the knee joint on MRI including cartilage lesions^{92,93}, meniscus tears^{93,94}, and ACL injuries^{94,95}, with diagnostic performance similar to experienced human readers. Recent studies have also used deep learning approaches for assigning a KL grade to assess the severity of joint degeneration on knee X-rays^{96,97}. A study by Pedoia *et al.* involving 4384 subjects from the OAI compared deep learning and traditional machine learning models for voxel based relaxometry analysis of cartilage T2 maps for distinguishing between knees with and without OA. The deep learning model had higher diagnostic performance with an AUC of 0.83 compared to an AUC of 0.78 for the traditional machine learning model⁹⁸.

연구 목적

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

연구 내용

1. Hyperparameter에 따른 정확도 비교

: 인공지능을 학습하기 위해 설정하는 초기값

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

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

연구 내용

2. CNN 구조에 따른 정확도 비교

- ✓CNN에서 하나의 Layer는 이미지의 특징을 추출하는 과정
- ✓Layer가 많다고 무작정 성능이 좋은 것만은 아님.
- ✓2개를 기준으로 잡고, 1개나 3개로 성능 비교

연구 내용

2. CNN 구조에 따른 정확도 비교

```
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
```

연구 내용

3. Optimizer의 종류에 따른 정확도 비교

- ✓*Loss function의 최솟값을 찾기 위해 사용하는 **최적화 알고리즘**
- ✓Stochastic gradient descent (SGD)가 대표적
- ✓SGD 외에도 momentum, Adagrad, Adam 등의 optimizer를 활용

*Loss function: 실제 답에 대한 인공지능이 도출한 답의 차이를 수식으로 표현한 것.
값이 작아지는 방향으로 학습.

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

연구 방법 및 일정

Google colaboratory

- Google이 제공하는 무료 오픈소스
- 컴퓨터에 python을 설치하지 않아도 인터넷으로 작동 가능
- Google drive와 연동 가능

Pytorch

- Facebook에서 개발한 python 라이브러리
- Dataset, data loader 및 transform, 모델 구성에 필요한 명령어

연구 방법 및 일정

Google colaboratory

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()
```

Pytorch

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')

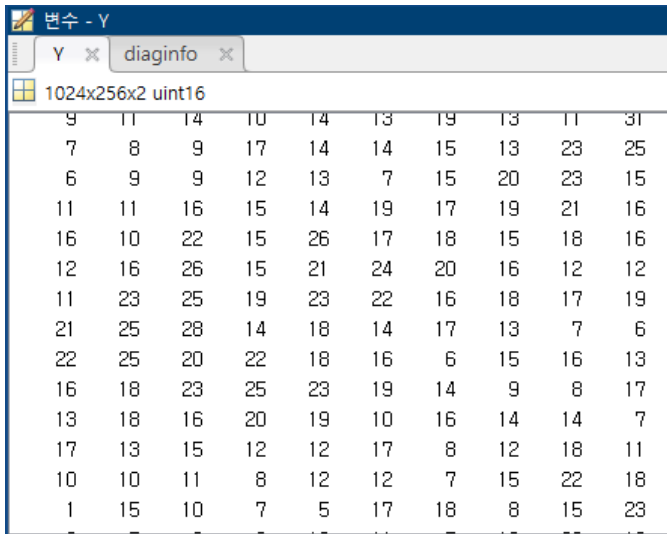
print('device:', device)

device: cpu
```

연구 방법 및 일정

Data set

- 핵의학과 전태주(강남세브란스), 정용휴(용인세브란스) 교수님
- 205명의 Bone scan 데이터 (Matlab)

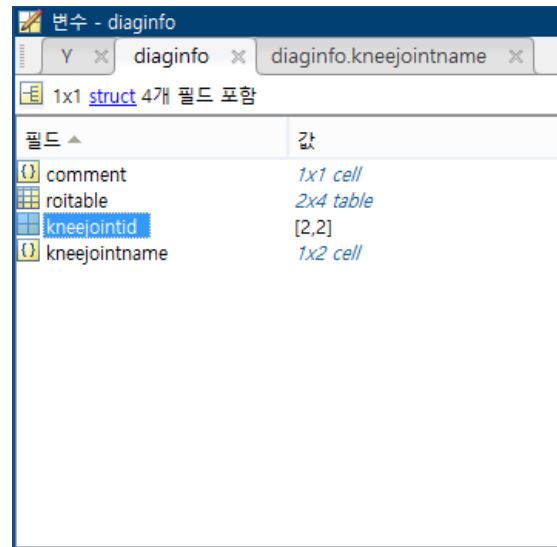


변수 - Y

Y x diaginfo x

1024x256x2 uint16

9	11	14	10	14	13	19	13	11	31
7	8	9	17	14	14	15	13	23	25
6	9	9	12	13	7	15	20	23	15
11	11	16	15	14	19	17	19	21	16
16	10	22	15	26	17	18	15	18	16
12	16	26	15	21	24	20	16	12	12
11	23	25	19	23	22	16	18	17	19
21	25	28	14	18	14	17	13	7	6
22	25	20	22	18	16	6	15	16	13
16	18	23	25	23	19	14	9	8	17
13	18	16	20	19	10	16	14	14	7
17	13	15	12	12	17	8	12	18	11
10	10	11	8	12	12	7	15	22	18
1	15	10	7	5	17	18	8	15	23

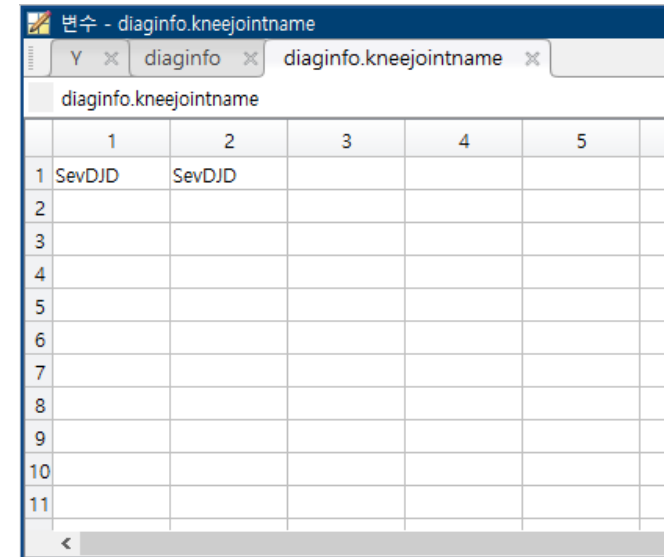


변수 - diaginfo

Y x diaginfo x diaginfo.kneejointname x

1x1 struct 4개 필드 포함

필드	값
comment	1x1 cell
roitable	2x4 table
kneejointid	[2,2]
kneejointname	1x2 cell



변수 - diaginfo.kneejointname

Y x diaginfo x diaginfo.kneejointname x

diaginfo.kneejointname

	1	2	3	4	5
1	SevDJD	SevDJD			
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

연구 방법 및 일정

Data set

- 205개의 1024*256*2 uint16 images
- Kneejointid: [R, L]
- Kneejointname: severity of DJD

ID	-1	2	3	4	5	6
Name	unspect	normal	minimal	mildDJD	modDJD	sevDJD

ex) [2, 5] = Right knee: normal / Left knee: modDJD

연구 방법 및 일정

Data set

- (우) Matlab data에 labeling을 하는 과정
- (하) 마지막 40개의 data는 test set으로 설정

```
test_data_set = []
test_label_set = []

for i in range(160,200):
    test_data_set.append(data_set_t[i])
    test_label_set.append(label_set[i])

test_data_set = np.array(test_data_set)
test_label_set = np.array(test_label_set)

test_label_set.dtype
```

```
Preview dataset

[88] data_set = []
    label_set = []
    num = 0

    for i in matlist[0:]:
        if (i == '0000001.mat') or (i == '0000058.mat') or (i == '0000615.mat') or (i == '0000823.mat') : continue
        mat_file = io.loadmat ('/content/drive/My_Drive/1/' + i)
        numbers = mat_file ['V']
        info = mat_file ['diaginfo']
        label = info['kneejointid'][0][0][0] - 2

        x1 = 510
        x2 = 638

        ## 1차 배치
        for j in range(100,150):
            if numbers[x1,j,0] == 0: break
            if j == 149:
                x1 = x1 + 70
                x2 = x2 + 70

        ## 2차 배치
        max_col = np.argmax(numbers[x1:x2,0:,0], axis = 1)
        max = []
        l = ...
```

```
for j in max_col:
    max.append(numbers[k,j,0])
    k+=1

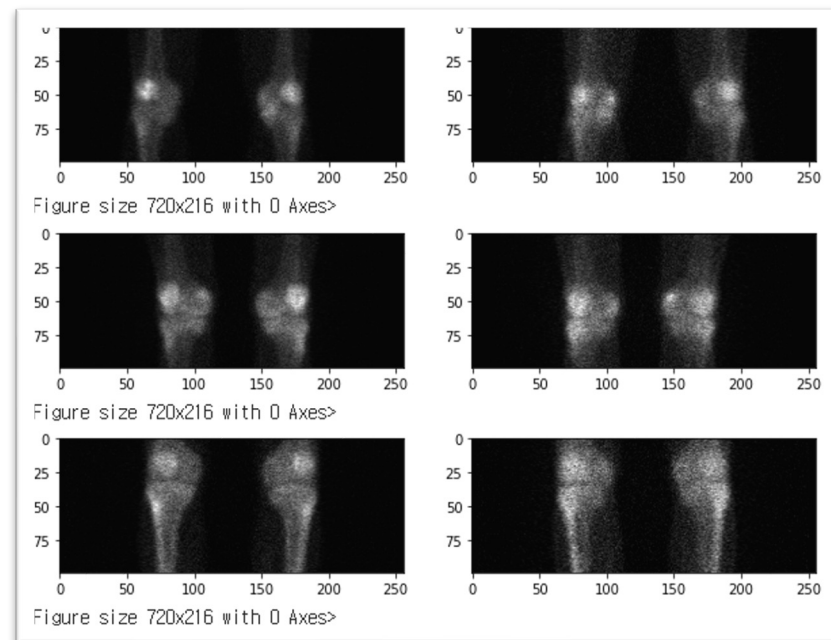
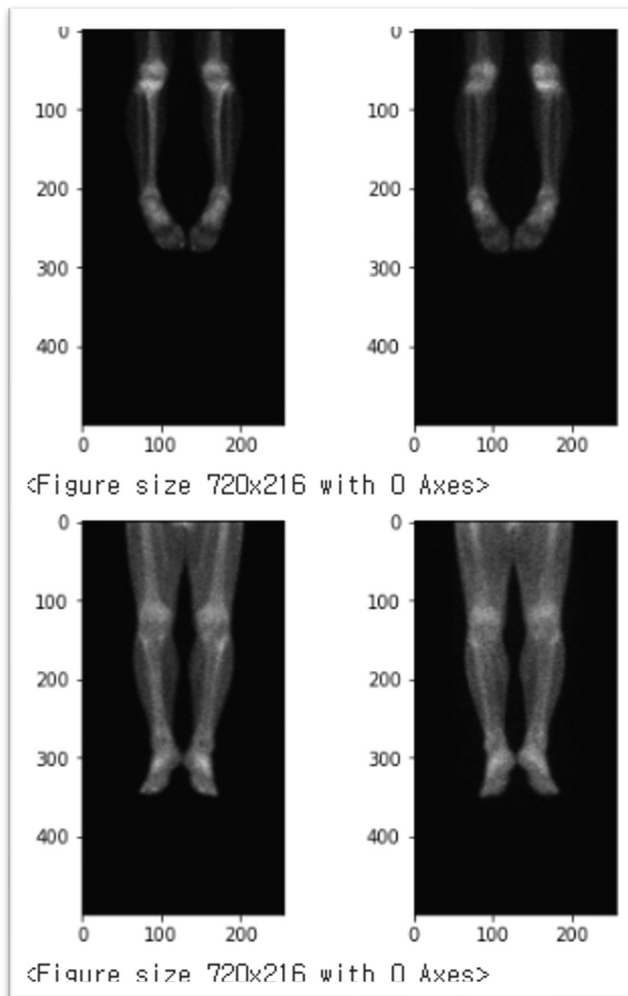
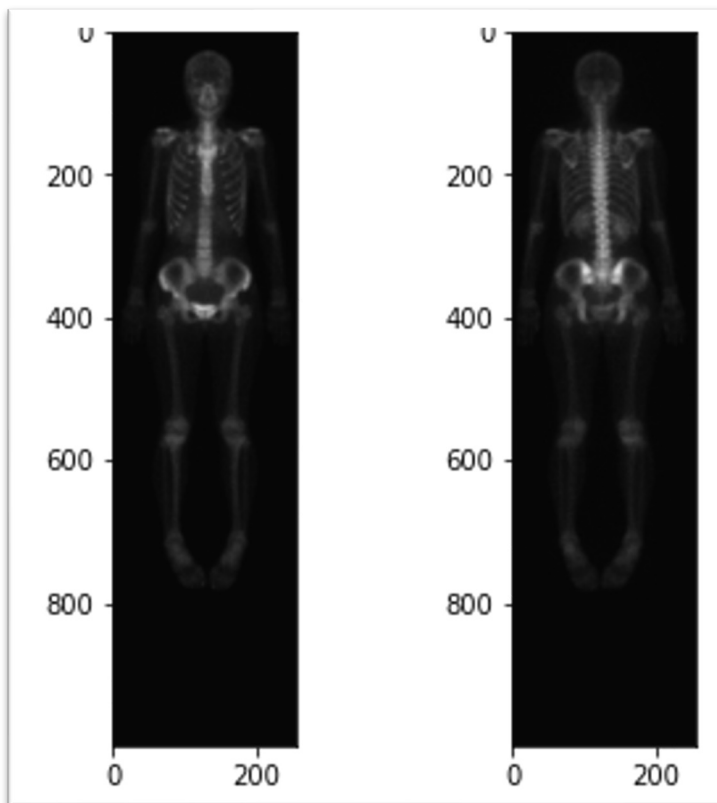
# 출력
x1 = np.argmax(max)+x1-64
x2 = np.argmax(max)+x2-64

data_set.append(numbers[x1:x2,0:128,0])
data_set.append(numbers[x1:x2,128:,0])
label_set.append(label[0])
label_set.append(label[1])

data_set = np.array(data_set, dtype = 'long')
data_set = data_set[np.newaxis, :, :, :]
label_set = np.array(label_set, dtype = 'long')
```

연구 방법 및 일정

Data set



연구 방법 및 일정

Model 학습

- Loss function, optimizer 설정
- Hyperparameter에 맞춰 학습 진행

```
#1 Loss: 1.6243 Acc: 18.9055% Time: 5.4354s
#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: 107.2290s
```

```
model = SimpleCNN()
model.to(device)

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

from torch.utils.data import DataLoader

train_dataloader = DataLoader(data_set, batch_size=batch_size, shuffle=True)
test_dataloader = DataLoader(test_data_set, batch_size=batch_size, shuffle=True)
```

```
UIJ 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.
```

기대 효과

- Bone scan 염증에 대한 높은 민감도
→ 초기 퇴행성 관절염 진단에 용이
- CNN은 이미지에서 특징을 추출하여 학습하는 알고리즘
→ 해상도가 낮은 핵의학 영상을 grade 별로 분류 가능