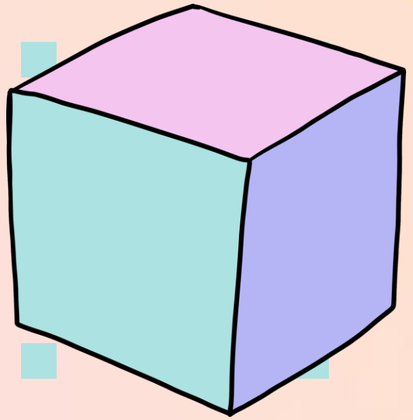
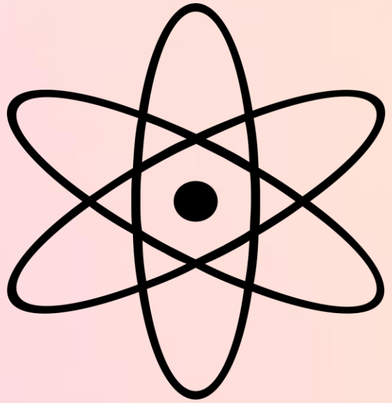


Gym Machine Classifier

Project B

전영준

YCS1003.03 2023-1



운동에 대한 관심 증가

운동이 트렌드, 취미화

운동기구 설명 부족

부상의 위험 및 PT 비용 부담

노인 인구의 매체 활용 저조

운동기구 앱

사진을 찍어 업로드하면

운동기구를 인식해서 설명

CNN모델을 이용한 운동 기구 분류기

1

데이터

1. Google Images Download

2. Images Crawling

Selenium
WebDriverManager
Beautiful Soup

2

데이터셋 구성
모델 구성

1. Torch Vision

2. CNN Model

3

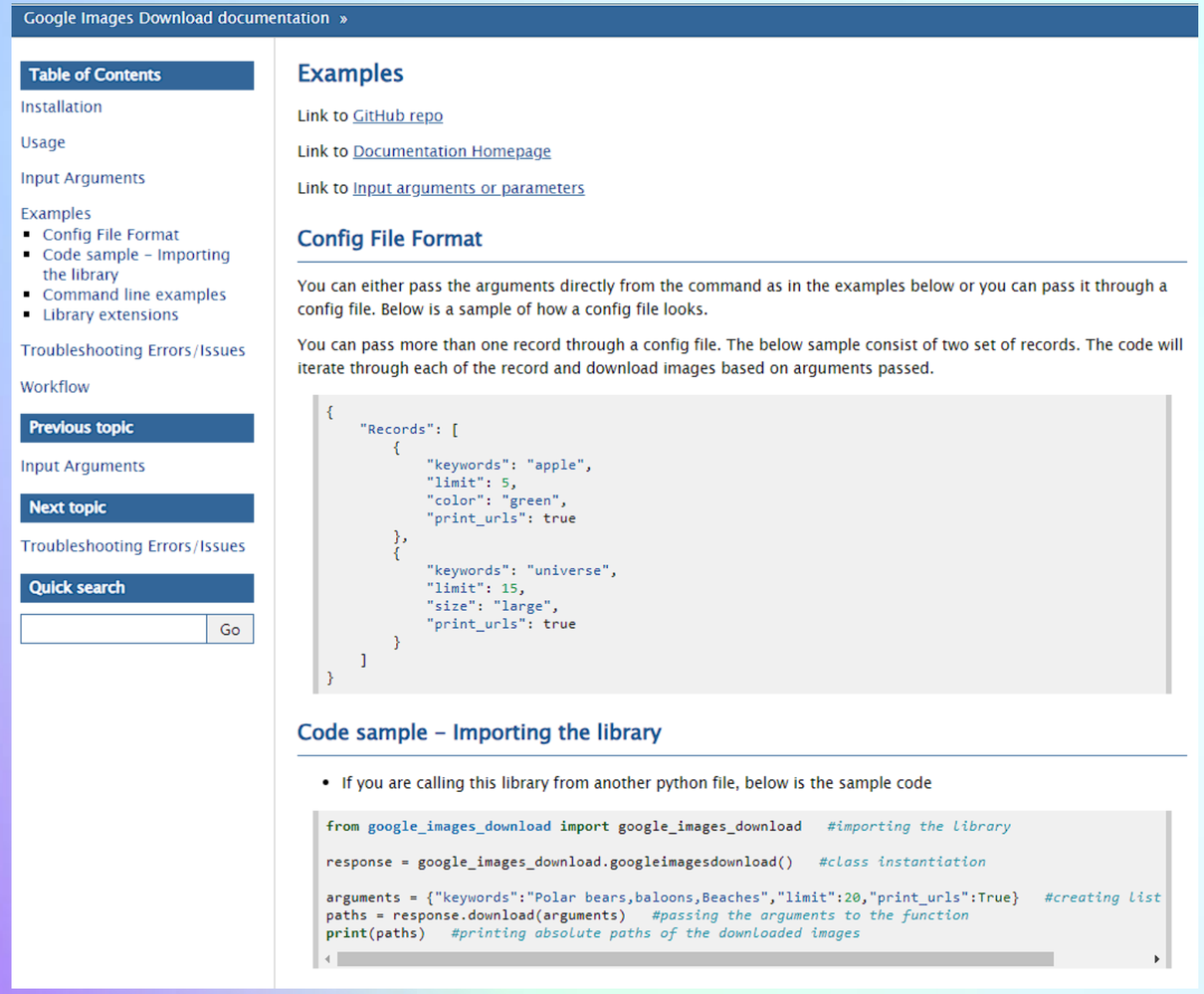
모델 비교를 통한 성능 향상

1. Channel 수

2. Fully Connected Layer 크기

데이터

1. Google Images Download



Google Images Download documentation »

Table of Contents

Installation

Usage

Input Arguments

Examples
- Config File Format
- Code sample – Importing the library
- Command line examples
- Library extensions

Troubleshooting Errors/Issues

Workflow

Previous topic

Input Arguments

Next topic

Troubleshooting Errors/Issues

Quick search

Go

Examples

Link to GitHub repo

Link to Documentation Homepage

Link to Input arguments or parameters

Config File Format

You can either pass the arguments directly from the command as in the examples below or you can pass it through a config file. Below is a sample of how a config file looks.

You can pass more than one record through a config file. The below sample consist of two set of records. The code will iterate through each of the record and download images based on arguments passed.

```
{
    "Records": [
        {
            "keywords": "apple",
            "limit": 5,
            "color": "green",
            "print_urls": true
        },
        {
            "keywords": "universe",
            "limit": 15,
            "size": "large",
            "print_urls": true
        }
    ]
}
```

Code sample – Importing the library

- If you are calling this library from another python file, below is the sample code

```
from google_images_download import google_images_download   #importing the library

response = google_images_download.googleimagesdownload()   #class instantiation

arguments = {"keywords":"Polar bears,baloons,Beaches","limit":20,"print_urls":True}   #creating list
paths = response.download(arguments)   #passing the arguments to the function
print(paths)   #printing absolute paths of the downloaded images
```

limit	I	Denotes number of images that you want to download. You can specify any integer value here. It will try and get all the images that it finds in the google image search page. If this value is not specified, it defaults to 100. **Note**: In case of occasional errors while downloading images, you could get less than 100 (if the limit is set to 100)

1. 장점
구글에서 지원
짧은 코드로 사진 다운 가능

2. 단점 & 사용하지 않은 이유
1개의 keyword로 최대 100장의 사진
다운로드를 실패하는 경우 다수

출처 https://google-images-download.readthedocs.io/en/latest/index.html

데이터

2. Images Crawling

1) 필요한 패키지 import

```
import urllib.request

from selenium import webdriver
from selenium.webdriver import Keys
from selenium.webdriver.common.by import By
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
from bs4 import BeautifulSoup
import time
```

2) 구글 이미지에 들어가서 특정 keyword를 검색하는 함수 정의

```python
def image_downloader(keyword):
    chrome_option = webdriver.ChromeOptions()
    chrome_option.add_experimental_option('detach', True)
    driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=chrome_option)

    driver.maximize_window()
    driver.get('https://www.google.co.kr/imghp?hl=ko&ogbl')
    time.sleep(2)

    search = driver.find_element(By.XPATH, '//*[@id="APjFqb"]')
    search.click()
    search.send_keys(keyword)
    search.send_keys(Keys.ENTER)
    time.sleep(2.3)
```

```
i = 0
while i < 40:
    driver.find_element(By.TAG_NAME, 'body').send_keys(Keys.PAGE_DOWN)
    time.sleep(2)
    i += 1

try:
    the_bo_gi = driver.find_element(By.CSS_SELECTOR, '#islmp > div > div > div > div > div.gBPM8 > div.qvfT1 > div.YstHxe > input')
    the_bo_gi.click()

except:
    continue

    i = 0
    while i < 30:
        driver.find_element(By.TAG_NAME, 'body').send_keys(Keys.PAGE_DOWN)
        time.sleep(2)
        i += 1

html_source = driver.page_source
soup = BeautifulSoup(html_source, 'html.parser')
images = soup.find_all('div', class_='bRMDJf islir')
k = 1
for img in images:
    img_url = img.find('img').get('src')
    try:
        urllib.request.urlretrieve(img_url, 'C:/images/' + keyword + '/' + str(k) + '.jpg')
        k += 1
        time.sleep(0.5)
    except:
        continue

    if k == 1000:
        print('end')
        break
```

스크롤을 최대로 내리고

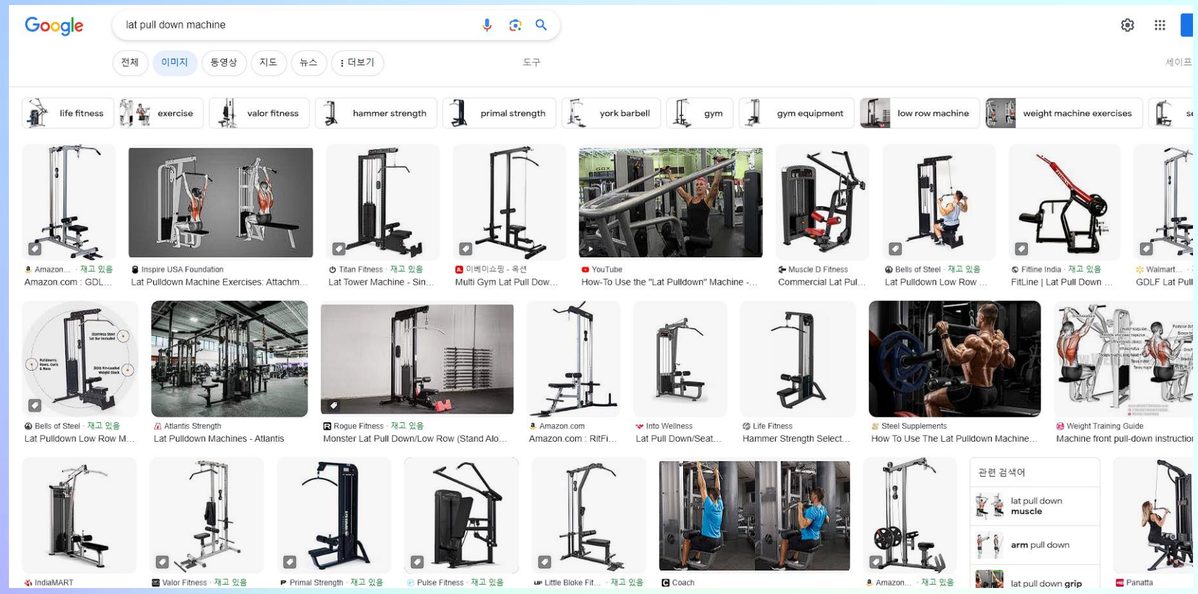
이미지의 Class를 모두 찾아서
이미지 url을 얻고

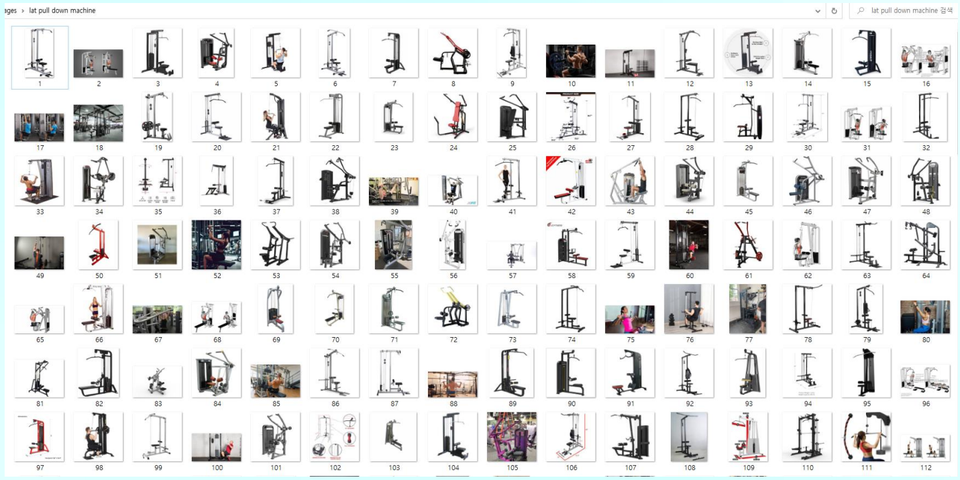
드라이브(C) 내의 폴더에
다운받도록 코드 구성

3) 키워드 전달하여 함수 실행

```
keywords = ['leg curl machine', 'leg extension machine', 'bench press bench', 'lat pull down machine', 'seated row machine', 'shoulder press machine', 'leg press machine', 'chest press machine', 'rowing machine', 'stepmill']

for keyword in keywords:
    image_downloader(keyword)
```

데이터 예시(lat pull down machine)





출처 https://www.google.com/search?q=lat+pull+down+machine&hl=ko&tbm=isch&source=hp&biw=1920

시도 1. Google Image 약 400장씩 사용

‘검색 결과 더보기’를 누르지 않으면 약 400장의 결과가 노출됨

시도 2에서는 모든 사진을 다운받도록 Crawling code를 수정하여 데이터 수를 늘림

데이터 개수(total 3958 images)

leg curl machine	leg extension machine	bench press bench	lat pull down machine	seated row machine	shoulder press machine	leg press machine	chest press machine	rowing machine	stepmill
400	400	400	399	397	395	400	395	400	372

시도 1 - 데이터셋 구성

```
from torchvision.datasets import ImageFolder

data_dir = "/content/drive/MyDrive/Colab Notebooks/인이활/project B/images(시도1)"

size = 64

transform = T.Compose([
    T.Resize((size, size)),
    T.RandomHorizontalFlip(),
    T.RandomCrop(size, padding=8),
    T.ToTensor(),
    T.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

dataset = ImageFolder(root=data_dir, transform=transform)

train_ratio = 0.8
train_size = int(train_ratio * len(dataset))
test_size = len(dataset) - train_size
train_set, test_set = torch.utils.data.random_split(dataset, [train_size, test_size])
```

10개 폴더에 나누어 담겨있는 이미지들을 데이터로 사용하기 위해 torchvisin.datasets의 ImageFolder 이용하였고,

64 x 64 크기의 이미지로 resize하여 각종 transform을 포함한 데이터셋 구성

시도 1 - 모델 구성

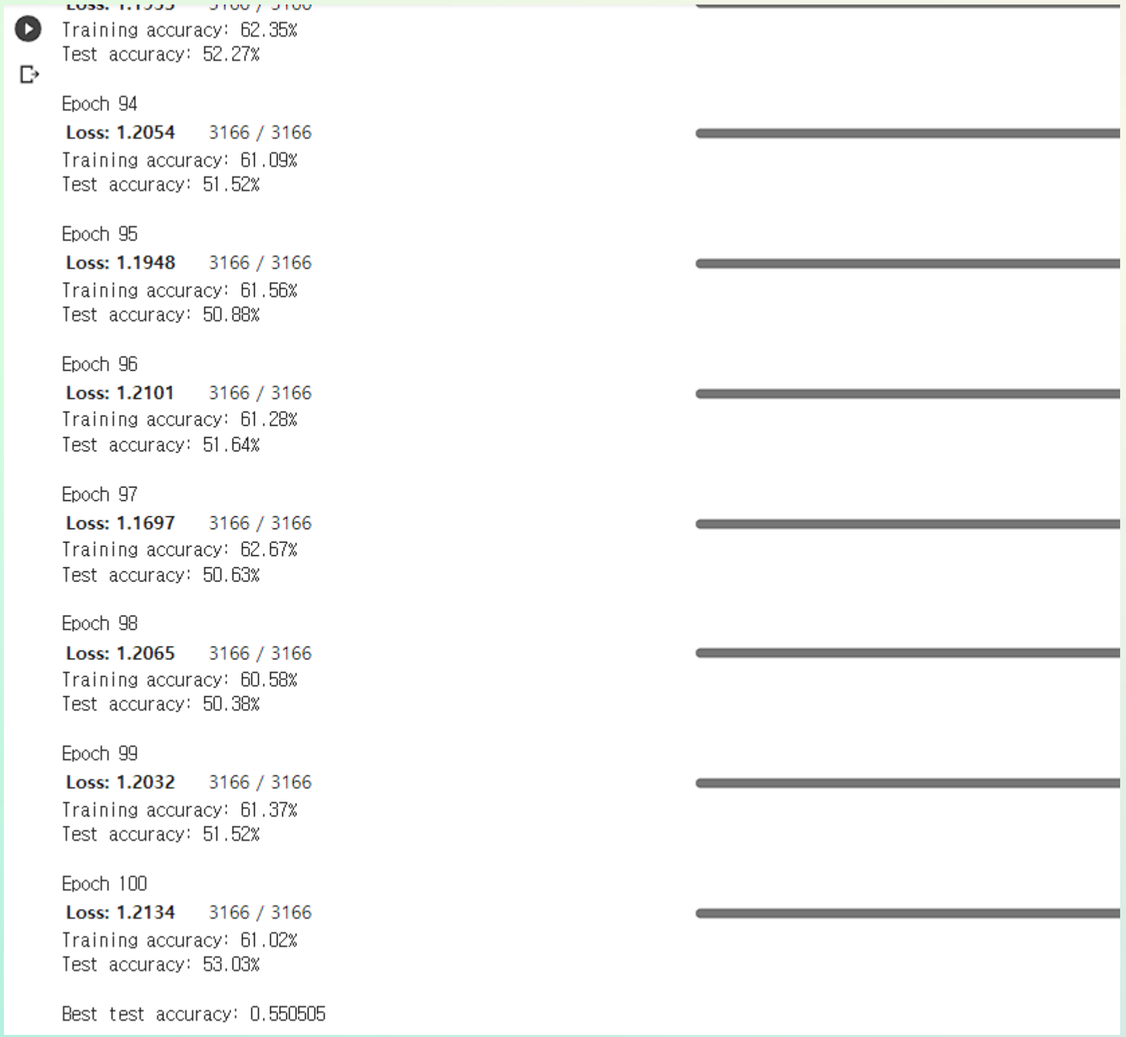
```
class CNN_CPCPCCPFF(nn.Module):

    def __init__(self, channel1=32, channel2=64, channel3=128, fc1=256, fc2=128):
        super().__init__()

        self.conv_layers = nn.Sequential(
            nn.Conv2d( in_channels=3, out_channels=channel1, kernel_size=3, padding=1),
            nn.BatchNorm2d(channel1),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Dropout(0.2),

            nn.Conv2d( in_channels=channel1, out_channels=channel2, kernel_size=3, padding=1),
            nn.BatchNorm2d(channel2),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Dropout(0.2),

            nn.Conv2d( in_channels=channel2, out_channels=channel3, kernel_size=3, padding=1),
            nn.BatchNorm2d(channel3),
            nn.ReLU(),
            nn.Conv2d( in_channels=channel3, out_channels=channel3, kernel_size=3, padding=1),
            nn.BatchNorm2d(channel3),
            nn.ReLU(),

            nn.MaxPool2d(2),
            nn.Dropout(0.2)
        )
        self.fc_layers = nn.Sequential(

            nn.Linear( 8*8*channel3, fc1 ),
            nn.BatchNorm1d(fc1),
            nn.ReLU(),
            nn.Dropout(0.2),

            nn.Linear(fc1, fc2),
            nn.BatchNorm1d(fc2),
            nn.ReLU(),
            nn.Dropout(0.2),

            nn.Linear(fc2, 10),
            nn.BatchNorm1d(10),
            nn.ReLU()
        )

    def forward(self, x):
        x = self.conv_layers(x)
        x = x.view( x.size(0), -1 )
        x = self.fc_layers(x)
        return x
```

Data가 많지 않기 때문에,
Pretrained Model을 사용하는 것 보다는
기존 모델의 구성을 차용하는 방식 사용

1. AlexNet의
   Conv + Pooling + Conv + Pooling
   + Conv + Conv +Pooling
   + Dense + Dense + Dense(output)
   구성을 차용

2. Pooling을 통해 image의 크기가 줄어듦에 따라
   channel 수가 증가하도록 구성

시도 1 - 테스트 모델 결과



Batch size 32,
Epochs 100,
Learning rate 0.001,
10 epoch마다 learning rate 가 0.33배,
NAdam optimizer를 사용하여,

Test accuracy가 가장 높았을 때를
저장하도록 적용

적은 Data임에도 55.0505% 정확도

시도 1 - 여러 모델 비교를 통한 성능 향상

```
learning_rate = 0.001
channel_set = [[32, 64, 128], [64, 128, 256]]
fc_set = [[256, 128], [512, 256]]
acc_model1 = []
acc_model2 = []
for chnls in channel_set:
    for fcs in fc_set:
        print(chnls, fcs)

        model1 = CNN_CPCPCCPFF(channel1=chnls[0], channel2=chnls[1], channel3=chnls[2], fc1=fcs[0], fc2=fcs[1])
        model1.to(device)
        criterion = nn.CrossEntropyLoss()
        optimizer = torch.optim.NAdam(model1.parameters(), lr=learning_rate)
        lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.33)
        acc = train_and_calculate_best_acc(optimizer, model1)
        acc_model1.append(acc)

        model2 = CNN_CCPCCPCCPFF(channel1=chnls[0], channel2=chnls[1], channel3=chnls[2], fc1=fcs[0], fc2=fcs[1])
        model2.to(device)
        criterion = nn.CrossEntropyLoss()
        optimizer = torch.optim.NAdam(model2.parameters(), lr=learning_rate)
        lr_scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.33)
        acc = train_and_calculate_best_acc(optimizer, model2)
        acc_model2.append(acc)

acc_model1, acc_model2
```

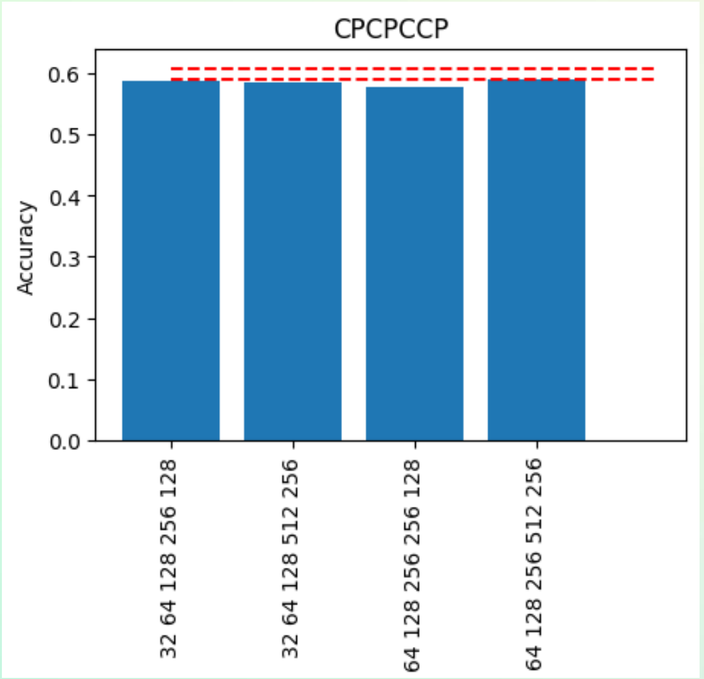
Conv + Pooling + Conv + Pooling
+ Conv + Conv +Pooling
+ Dense + Dense + Dense(output)

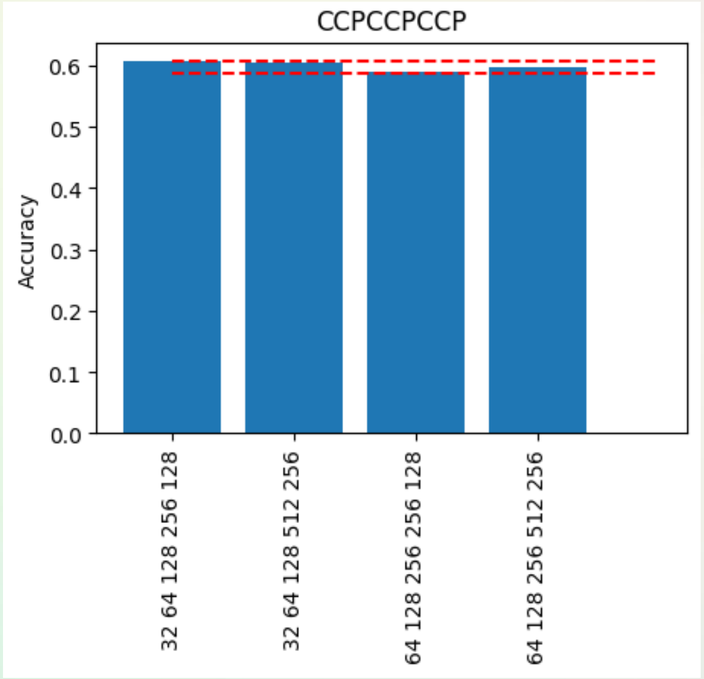
의 구조을 유지한 model 1과
Conv layer를 추가한 model 2

Channel 수와 Fully Connected Layer 크기를 조절하여 위 구조마다 4개 다른 모델을 구성(총 8개의 모델)

모델별로 50 epoch만 학습시켜 Accuracy를 비교

시도 1 - 여러 모델 비교를 통한 성능 향상





Conv + Conv + Pooling
+ Conv + Conv + Pooling
+ Conv + Conv + Pooling
+ Dense + Dense + Dense(output)

의 구조인 model 2가 성능이
대체적으로 좋음

가장 성능이 좋은 모델은 채널 수가 32, 32, 64, 64, 128, 128
/ fully connected layer 크기가 256, 128인 모델이고,
50 epoch 진행 후
정확도는 약 60.7323%

데이터 추가 확보 방법

```
i = 0
while i < 40:
    driver.find_element(By.TAG_NAME, 'body').send_keys(Keys.PAGE_DOWN)
    time.sleep(2)
    i += 1

try:
    the_bo_gi = driver.find_element(By.CSS_SELECTOR, '#islmp > div > div > div > div > div.gBPM8 > div.qvfT1 > div.YstHxe > input')
    the_bo_gi.click()

except:
    continue

    i = 0
    while i < 30:
        driver.find_element(By.TAG_NAME, 'body').send_keys(Keys.PAGE_DOWN)
        time.sleep(2)
        i += 1

html_source = driver.page_source
soup = BeautifulSoup(html_source, 'html.parser')
images = soup.find_all('div', class_='bRMDJf islir')
k = 1
for img in images:
    img_url = img.find('img').get('src')
    try:
        urllib.request.urlretrieve(img_url, 'C:/images/' + keyword + '/' + str(k) + '.jpg')
        k += 1
        time.sleep(0.5)
    except:
        continue

    if k == 1000:
        print('end')
        break
```

스크롤만 내려서 노출되는 사진은
400장 내로, 데이터 수가 적음

‘검색 결과 더보기’ 버튼을 클릭해서
모든 사진을 노출되도록 함

시도 2. Google Image 결과 모두 사용

모든 사진을 다운받도록 Crawling code를 수정하여 데이터 수를 늘림

종류별로 1000장 이상씩 데이터를 확보하는 것이 목표였으나,
검색 결과가 그에 미치지는 못했음

데이터 개수(total 5136 images)

leg curl machine	leg extension machine	bench press bench	lat pull down machine	seated row machine	shoulder press machine	leg press machine	chest press machine	rowing machine	stepmill
539	548	464	495	462	538	569	570	579	372

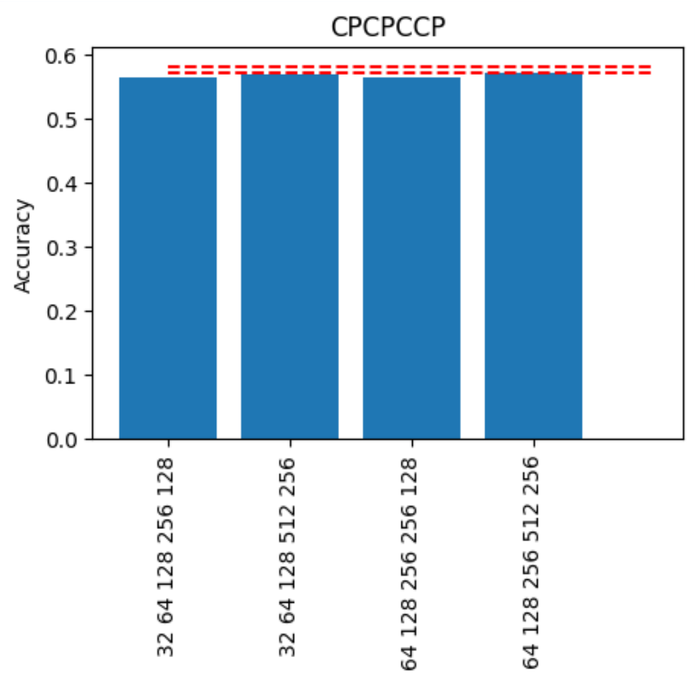
시도 2 - 테스트 모델 결과

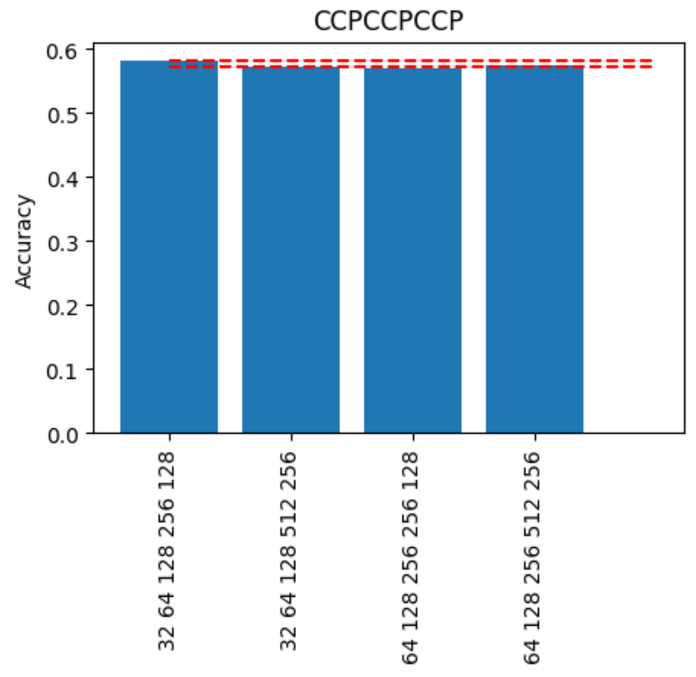
Batch size 32,
Epochs 100,
Learning rate 0.001,
10 epoch마다 learning rate 가 0.33배,
NAdam optimizer를 사용하여,

Test accuracy가 가장 높았을 때를
저장하도록 적용

적은 Data임에도 56.5175% 정확도

시도 2 - 여러 모델 비교를 통한 성능 향상

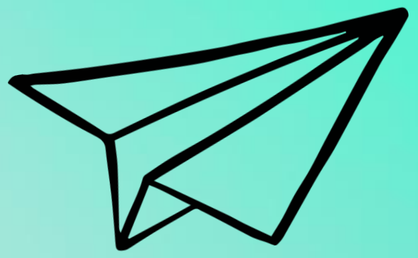




Conv + Conv + Pooling
+ Conv + Conv + Pooling
+ Conv + Conv + Pooling
+ Dense + Dense + Dense(output)

의 구조인 model 2가 성능이
대체적으로 좋음

가장 성능이 좋은 모델은 채널 수가 32, 32, 64, 64, 128, 128
/ fully connected layer 크기가 256, 128인 모델이고,
50 epoch 진행 후
정확도는 약 58.1712% / test 데이터가 소규모로, 편차가 클 것으로 생각하면 시도 1과 유사



감사합니다