**02. Selenium**

**CODE**

1) Crawling

def main():

    crawling()

    saveCSV()

### 1. Crawling

def crawling():

    file\_path = './2022.12/12.09\_d48\_image/data/selenium'

    chromedriver\_path = './2022.12/chromedriver.exe'

    os.makedirs(file\_path, exist\_ok=True)

    N = 200  # 데이터 수집 개수

    SEARCH\_LIST = ['T-Shirt', "Trouser", "Dress", "Bag", "Sandal"]

    driver = webdriver.Chrome(chromedriver\_path)

    driver.implicitly\_wait(10)

    for search in SEARCH\_LIST:

        img\_dir = os.path.join(file\_path, search)

        os.makedirs(img\_dir, exist\_ok=True)

        driver.get('https://www.google.com')

        elem = driver.find\_element(By.NAME, 'q')

        elem.clear()

        elem.send\_keys(search)          # 검색

        elem.send\_keys(Keys.RETURN)     # 엔터

        assert "No results found." not in driver.page\_source

        # 이미지 메뉴 누르기

        driver.find\_element(By.XPATH, '//\*[@id="hdtb-msb"]/div[1]/div/div[2]/a').click()

        selenium\_scroll\_option(driver)

        img\_srcs = driver.find\_elements(By.CLASS\_NAME, 'rg\_i')

        url\_list = []

        last = 0

        for idx, img\_src in enumerate(img\_srcs):

            base64\_image = img\_src.get\_attribute('src')

            try:

                if base64\_image:  # 64일 때

                    if 'base64' in base64\_image:

                        img = Image.open(BytesIO(base64.b64decode(base64\_image.split(',')[-1])))  # decode 후 저장

                        img.save(os.path.join(img\_dir, search+str(idx)+'.png'))

                        last = idx + 1

                    else:

                        url\_list.append(base64\_image)

            except PIL.UnidentifiedImageError:

                print(base64\_image)

            except AttributeError:

                print(base64\_image)

                continue

            if int(N) == idx:

                break

        [urllib.request.urlretrieve(url, os.path.join(img\_dir, search+str(last+i)+'.png')) for i, url in enumerate(set(url\_list))]

    driver.close()

2) Resize and Save CSV file

#### 2. Image Reszie & Save CSV file

def saveCSV():

    SEARCH\_LIST = ['T-Shirt', "Trouser", "Dress", "Bag", "Sandal"]

    TRAIN\_PERCENTAGE = 0.8  # training 비율

    IMAGE\_SIZE = 28         # 이미지 사이즈

    file\_path = './2022.12/12.09\_d48\_image/data/selenium'

    df\_dict\_train= {

    'file\_name':[],

    'label':[]

    }

    df\_dict\_test= {

    'file\_name':[],

    'label':[]

    }

    os.makedirs(os.path.join(file\_path, 'train'), exist\_ok=True)

    os.makedirs(os.path.join(file\_path, 'test') , exist\_ok=True)

    for idx in range(len(SEARCH\_LIST)):

        img\_dir = os.path.join(file\_path, SEARCH\_LIST[idx])  # image file path

        for index, item in enumerate(os.listdir(img\_dir)):

            image = cv2.imread(os.path.join(img\_dir,item), cv2.IMREAD\_GRAYSCALE)    # gray scale

            image = cv2.resize(image,(IMAGE\_SIZE,IMAGE\_SIZE))                       # resize

            if int(index) < len(os.listdir(img\_dir)) \* TRAIN\_PERCENTAGE :   # train 비율 80%

                cv2.imwrite( os.path.join(file\_path, 'train', item), image)

                df\_dict\_train['file\_name'].append(item)

                df\_dict\_train['label'].append(idx)

            else:                                                           # test  비율 20%

                cv2.imwrite( os.path.join(file\_path, 'test', item), image)

                df\_dict\_test['file\_name'].append(item)

                df\_dict\_test['label'].append(idx)

        df\_train = pd.DataFrame(df\_dict\_train)

        df\_test  = pd.DataFrame(df\_dict\_test)

    df\_train.to\_csv(os.path.join(file\_path, 'annotation\_train.csv'))

    df\_test.to\_csv (os.path.join(file\_path, 'annotation\_test.csv'))

if \_\_name\_\_ =='\_\_main\_\_':

    main()

3) Train & Test

file\_path = './2022.12/12.09\_d48\_image/data/selenium'

IMAGE\_SIZE = 28

class CustomImageDataset(Dataset):

    def \_\_init\_\_(self, annotations\_file, img\_dir, transform=None, target\_transform=None):

        self.img\_labels = pd.read\_csv(annotations\_file, names=['file\_name', 'label'], skiprows=[0])

        self.img\_dir = img\_dir

        self.transform = transform

        self.target\_transform = target\_transform

    def \_\_len\_\_(self):

        return len(self.img\_labels)

    def \_\_getitem\_\_(self, idx):

        img\_path = os.path.join(self.img\_dir, self.img\_labels.iloc[idx, 0])

        try:

            image = read\_image(img\_path)

        except:

            print(self.img\_labels.iloc[idx, 0])

            exit()

        label = int(self.img\_labels.iloc[idx, 1])

        if self.transform:

            image = self.transform(image)

        if self.target\_transform:

            label = self.target\_transform(label)

        return image, label

#### Define Neural Netowrk model

class Net(nn.Module):

    def \_\_init\_\_(self):

        super(Net, self).\_\_init\_\_()

        self.fc1 = nn.Linear(784, 512)  # input 28x28 = 784

        self.fc2 = nn.Linear(512, 256)

        self.fc3 = nn.Linear(256, 128)

        self.fc4 = nn.Linear(128, 64)

        self.fc5 = nn.Linear(64, 32)

        self.fc6 = nn.Linear(32, 10)    # Output 0~9 = 10 labels

    def forward(self, x):

        x = x.float()

        h1 = F.relu(self.fc1(x.view(-1, 784)))

        h2 = F.relu(self.fc2(h1))

        h3 = F.relu(self.fc3(h2))

        h4 = F.relu(self.fc4(h3))

        h5 = F.relu(self.fc5(h4))

        h6 = self.fc6(h5)

        return F.log\_softmax(h6, dim=1)

#### Prepare Data Loader for Training and Validation

transform = transforms.Compose([

                        transforms.ToTensor(),

                        transforms.Normalize((0.1307,), (0.3081,))

                        ])

#### vars and deivce 설정

epochs = 10         # 몇번 학습

lr = 0.001          # learning rate

momentum = 0.5      # optimizer 최적화 함수에 들어가는 관성계수

no\_cuda = True      # cuda인지

seed = 1            # random seed

log\_interval = 5

use\_cuda = not no\_cuda and torch.cuda.is\_available()

torch.manual\_seed(seed)

# cuda면 cuda쓰고 아니면 cpu 사용

device = torch.device("cuda" if use\_cuda else "cpu")

kwargs = {'num\_workers': 1, 'pin\_memory': True} if use\_cuda else{}

                # 1: 사용 프로세서 1로설정

batch\_size = 16    # 한번 학습할때 몇개

test\_batch\_size = 16

dataset\_train = CustomImageDataset(

    annotations\_file= os.path.join(file\_path, 'annotation\_train.csv'),

    img\_dir= file\_path + '/train',

    )

dataset\_test = CustomImageDataset(

    annotations\_file= os.path.join(file\_path, 'annotation\_test.csv'),

    img\_dir= file\_path + '/test',

    )

train\_loader = torch.utils.data.DataLoader(dataset\_train, batch\_size= batch\_size, shuffle=True)

test\_loader = torch.utils.data.DataLoader(dataset\_test, batch\_size=test\_batch\_size, shuffle=True, \*\*kwargs)

model = Net().to(device)  # devce : cpu or gpu ?? 나는 cpu

optimizer = optim.SGD(model.parameters(), lr= lr, momentum= momentum)

def train(log\_interval, model, device, train\_loader, optimizer, epoch):

    model.train()

    for batch\_idx, (data, target) in enumerate(train\_loader):

        data, target = data.to(device), target.to(device)

        output = model(data)

        loss = F.nll\_loss(output, target)

        loss.backward()

        optimizer.step()

        if batch\_idx % log\_interval == 0:

            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(

                epoch, batch\_idx \* len(data), len(train\_loader.dataset),

                100. \* batch\_idx / len(train\_loader), loss.item()))

def test(log\_interval, model, device, test\_loader):

    model.eval()

    test\_loss = 0

    correct = 0

    with torch.no\_grad():

        for data, target in test\_loader:

            data, target = data.to(device), target.to(device)

            output = model(data)

            test\_loss += F.nll\_loss(output, target, reduction='sum').item()

            pred = output.argmax(dim=1, keepdim=True)

            correct += pred.eq(target.view\_as(pred)).sum().item()

    test\_loss /= len(test\_loader.dataset)

    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format

          (test\_loss, correct, len(test\_loader.dataset),

        100. \* correct / len(test\_loader.dataset)))

for epoch in range(1, epochs+1):

    print(epoch)

    train(log\_interval, model, device, train\_loader, optimizer, epoch)

test(log\_interval, model, device, test\_loader)

torch.save(model, file\_path + '/model.pt')  # 가중치

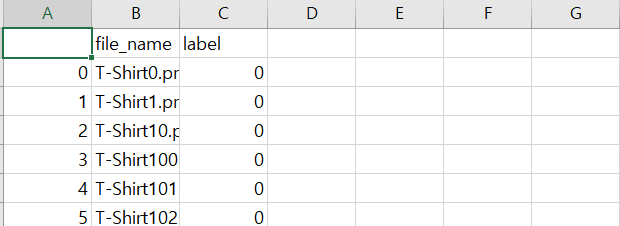
**RESULT**

1) File List

테이블이(가) 표시된 사진

자동 생성된 설명

2) Trainset (794개)



* T-Shirt 160개, Trouser 161개, Dress 160개, Bag 157개, Sandal 156개

3) Testset (197개)

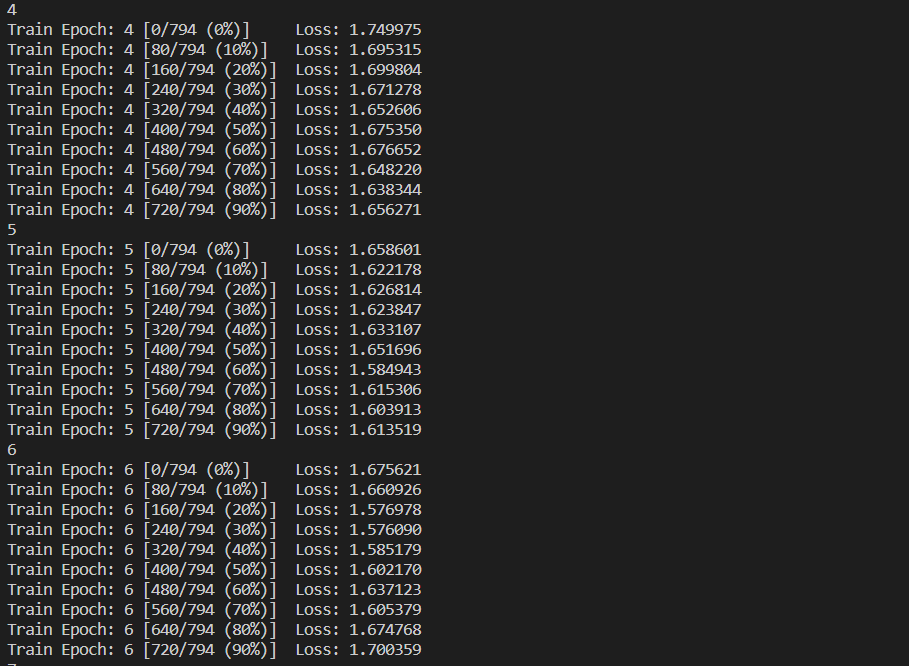
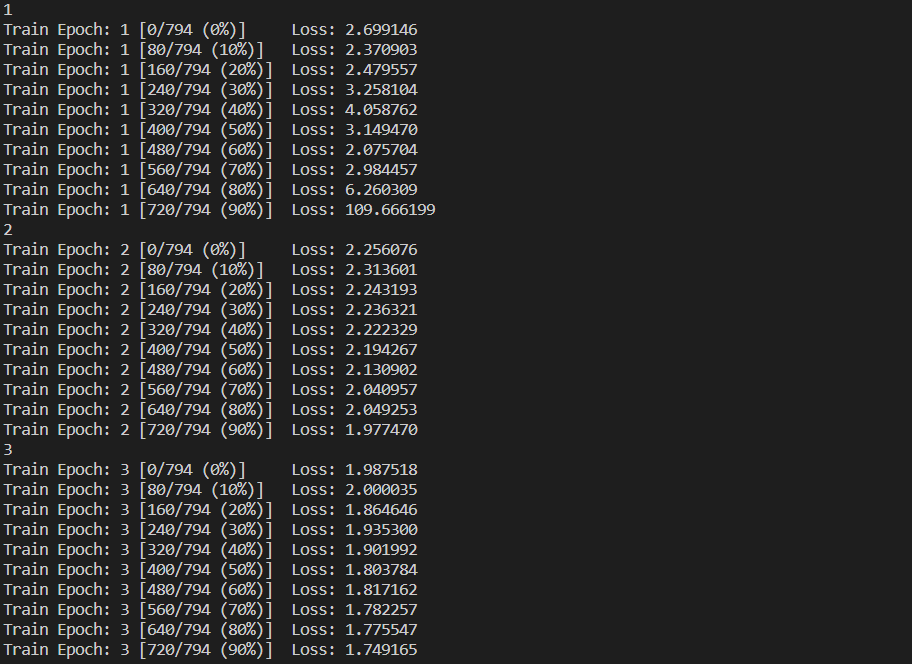
테이블이(가) 표시된 사진

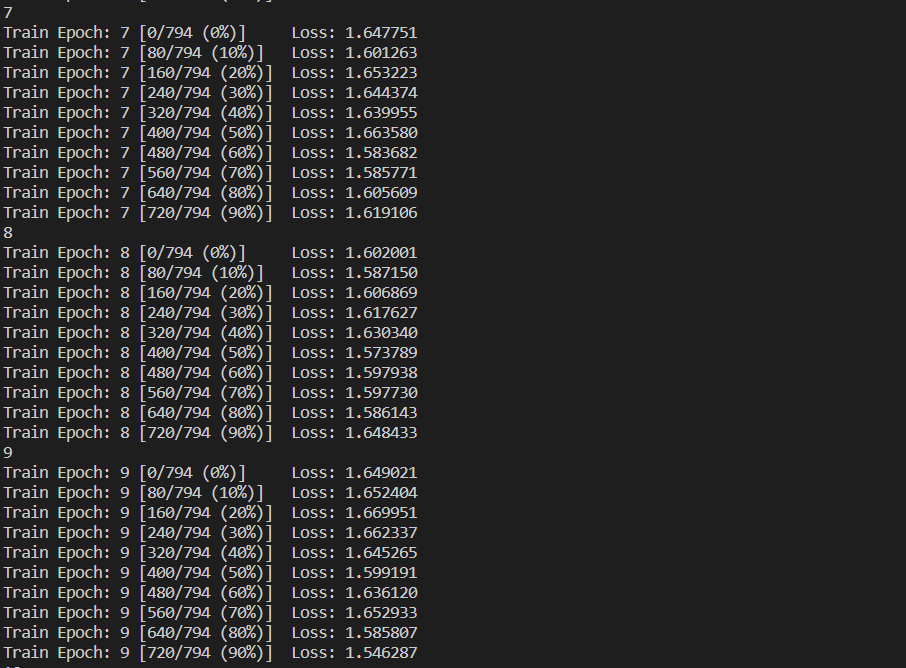
자동 생성된 설명

* T-Shirt 40개, Trouser 40개, Dress 40개, Bag 39개, Sandal 38개

4) Train & Test

- Train



텍스트이(가) 표시된 사진

자동 생성된 설명

- Test



정확도는 20%로 그냥 찍는 정도입니다….