프로젝트 결과보고서

김정현

1. 프로젝트 구현 결과

```
model.eval()
start_time = time.time()

with torch.no_grad():
    running_loss = 0.
    running_corrects = 0

    for inputs, labels in test_dataloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

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

        # 한 배치의 첫 번째 이미지에 대하여 결과 시각화
        print(f'[예측 결과: {class_names[preds[0]]}] (실제 정답: {class_names[labels.data[0]]})')
        imshow(inputs.cpu().data[0], title='예측 결과: ' + class_names[preds[0]])

    epoch_loss = running_loss / len(test_datasets)
    epoch_acc = running_corrects / len(test_datasets) * 100.
    print('[Test Phase] Loss: {:.4f} Acc: {:.4f}% Time: {:.4f}s'.format(epoch_loss, epoch_acc, time.time() - start_time))
```

학습된 모델평가

학습된 모델평가 단계

이미지 시각화로 얻은
정확도 높은 결과

이미지 시각화로 얻은 결과

- 제작품 설명

이 프로젝트는 한국 남자배우의 얼굴을 구별하는 이미지 분류 모델을 구현하고 평가하는 것을 목표로 합니다. 프로젝트에서는 전이 학습(Transfer Learning)을 활용하여 이미지 분류 모델을 학습시키고 학습된 모델을 사용하여 주어진 한국 남자배우의 얼굴 이미지를 받고, 해당 배우의 식별 결과를 예측하는 것이 프로젝트의 최종 목표입니다.
모델을 만들기 위해서 Python을 위한 오픈소스 머신 러닝 라이브러리인 PyTorch와 torchvision 등의 라이브러리를 사용하였고, 단계에는 이미지 데이터 수집, 전처리, 모델 구성, 학습 과정, 평가 과정으로 구성하였습니다.
우선 bing_image_downloader 라이브러리를 사용하여 총 10명의 한국 남자배우를 검색하여 지정하고 관련 이미지를 크롤링 하였고 수집한 이미지는 학습 데이터와 평가 데이터로 구분하여 관리하였습니다.
다음으로는 수집한 이미지 데이터를 학습에 활용하기 위해 전처리 과정을 거쳤습니다. 이 과정에서는 torchvision.transforms를 사용하여 이미지 크기를 조정하고 이미지를 텐서로 변환하고 정규화 작업을 수행하여 모델의 입력에 적합하게 처리했습니다.
또한, torchvision에서 제공하는 사전 학습된 모델로, 이미지 분류에 널리 사용되는 모델인 ResNet-34 모델을 사용합니다. ResNet-34의 Fully Connected 레이어를 원하는 출력 뉴런 수에 맞게 변경해서 사용하여 한국 남자배우들을 분류할 수 있는 모델을 구성하였습니다.
모델 구성 후에는 학습 및 평가를 수행합니다. 학습 데이터를 배치 단위로 불러와 모델에 입력하고, 역전파를 통해 기울기 계산 및 학습을 진행하여 모델을 학습 시킨 뒤 손실(loss)과 정확도(accuracy)를 계산하고 출력하여 모델의 성능을 평가합니다.
입력 이미지에 대한 모델의 예측 결과와 실제 정답을 비교하여 정확도를 계산하였고, 평가 과정에서는 예측 결과와 실제 정답을 시각적으로 비교하고 출력하여 모델의 분류 성능을 시각화하였습니다.

- 자료

수집한 이미지를 저장하기 위한 폴더를 생성하고, 필요한 함수를 정의합니다.

```
import os
import shutil
from bing_image_downloader.bing_image_downloader import downloader


directory_list = [
    './custom_dataset/train/',
    './custom_dataset/test/',
]

# 초기 디렉토리 만들기
for directory in directory_list:
    if not os.path.isdir(directory):
        os.makedirs(directory)

# 수집한 이미지를 학습 데이터와 평가 데이터로 구분하는 함수
def dataset_split(query, train_cnt):
    # 학습 및 평가 데이터셋 디렉토리 만들기
    for directory in directory_list:
        if not os.path.isdir(directory + '/' + query):
            os.makedirs(directory + '/' + query)
    # 학습 및 평가 데이터셋 준비하기
    cnt = 0
    for file_name in os.listdir(query):
        if cnt < train_cnt:
            print(f'[Train Dataset] {file_name}')
            shutil.move(query + '/' + file_name, './custom_dataset/train/' + query + '/' + file_name)
        else:
            print(f'[Test Dataset] {file_name}')
            shutil.move(query + '/' + file_name, './custom_dataset/test/' + query + '/' + file_name)
        cnt += 1
    shutil.rmtree(query)
```

```
model = models.resnet34(pretrained=True)
num_features = model.fc.in_features
# 전이 학습 : 모델의 출력 뉴런 수를 10개로 교체하여 마지막 레이어 다시 학습
model.fc = nn.Linear(num_features, 10)
model = model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

```
# 데이터셋을 불러올 때 사용할 객체 정의
transforms_train = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(), # 데이터 증진
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) # 정규화
])

transforms_test = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

data_dir = './custom_dataset'
train_datasets = datasets.ImageFolder(os.path.join(data_dir, 'train'), transforms_train)
test_datasets = datasets.ImageFolder(os.path.join(data_dir, 'test'), transforms_test)

train_dataloader = torch.utils.data.DataLoader(train_datasets, batch_size=4, shuffle=True, num_workers=4)
test_dataloader = torch.utils.data.DataLoader(test_datasets, batch_size=4, shuffle=True, num_workers=4)

print('학습 데이터셋 크기:', len(train_datasets))
print('테스트 데이터셋 크기:', len(test_datasets))

class_names = train_datasets.classes
print('클래스:', class_names)
```


```
num_epochs = 50
model.train()
start_time = time.time()

# 전체 반복(epoch) 수 만큼 반복
for epoch in range(num_epochs):
    running_loss = 0.
    running_corrects = 0

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

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

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

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

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

    # 학습 과정 중에 결과 출력
    print('#{} Loss: {:.4f} Acc: {:.4f}% Time: {:.4f}s'.format(epoch, epoch_loss, epoch_acc, time.time() - start_time))
```

```
#0 Loss: 2.3190 Acc: 19.6667% Time: 14.6027s
#1 Loss: 1.5697 Acc: 48.3333% Time: 22.9041s
#2 Loss: 1.0281 Acc: 66.3333% Time: 31.8019s
#3 Loss: 0.6332 Acc: 80.3333% Time: 39.4400s
#4 Loss: 0.4672 Acc: 84.6667% Time: 48.4028s
#5 Loss: 0.5104 Acc: 83.3333% Time: 57.6567s
#6 Loss: 0.4029 Acc: 87.6667% Time: 65.3342s
#7 Loss: 0.3361 Acc: 89.6667% Time: 74.4266s
#8 Loss: 0.2250 Acc: 93.6667% Time: 83.4512s
#9 Loss: 0.1954 Acc: 94.3333% Time: 91.0000s
#10 Loss: 0.2075 Acc: 92.0000% Time: 99.9434s
#11 Loss: 0.2205 Acc: 93.3333% Time: 109.0707s
#12 Loss: 0.2076 Acc: 93.3333% Time: 116.4249s
#13 Loss: 0.1501 Acc: 95.3333% Time: 125.5311s
#14 Loss: 0.2675 Acc: 93.6667% Time: 133.4729s
#15 Loss: 0.1647 Acc: 94.6667% Time: 142.2289s
#16 Loss: 0.2954 Acc: 91.3333% Time: 151.2845s
#17 Loss: 0.1522 Acc: 96.0000% Time: 158.8990s
#18 Loss: 0.1820 Acc: 94.6667% Time: 167.9218s
#19 Loss: 0.1199 Acc: 96.3333% Time: 176.9334s
#20 Loss: 0.1085 Acc: 97.0000% Time: 184.5459s
#21 Loss: 0.1528 Acc: 96.0000% Time: 193.5329s
#22 Loss: 0.1130 Acc: 96.6667% Time: 202.1689s
#23 Loss: 0.1402 Acc: 95.6667% Time: 209.9811s
#24 Loss: 0.1569 Acc: 94.3333% Time: 218.8252s
#25 Loss: 0.1360 Acc: 97.0000% Time: 226.7117s
#26 Loss: 0.1285 Acc: 97.3333% Time: 235.3841s
#27 Loss: 0.1431 Acc: 95.6667% Time: 244.4008s
#28 Loss: 0.0679 Acc: 99.0000% Time: 251.9775s
#29 Loss: 0.0820 Acc: 98.0000% Time: 261.0517s
#30 Loss: 0.1079 Acc: 97.3333% Time: 270.0046s
#31 Loss: 0.1057 Acc: 97.0000% Time: 277.7111s
#32 Loss: 0.0562 Acc: 98.6667% Time: 286.8275s
#33 Loss: 0.0545 Acc: 98.6667% Time: 295.7046s
#34 Loss: 0.0830 Acc: 98.6667% Time: 303.4542s
#35 Loss: 0.0535 Acc: 98.3333% Time: 312.4809s
#36 Loss: 0.0230 Acc: 100.0000% Time: 320.5181s
#37 Loss: 0.0673 Acc: 98.0000% Time: 328.7342s
#38 Loss: 0.0300 Acc: 99.0000% Time: 337.6665s
#39 Loss: 0.1063 Acc: 97.3333% Time: 345.2374s
#40 Loss: 0.1446 Acc: 96.0000% Time: 354.2518s
#41 Loss: 0.0758 Acc: 97.6667% Time: 363.4366s
#42 Loss: 0.0471 Acc: 98.3333% Time: 371.0586s
#43 Loss: 0.0929 Acc: 98.0000% Time: 380.0453s
#44 Loss: 0.0805 Acc: 97.6667% Time: 389.0690s
#45 Loss: 0.0759 Acc: 98.6667% Time: 396.7333s
```

2. 결론

프로젝트명	남자배우얼굴인식	이름	김정현

1. 개요

전이학습(Transfer Learning)과 이미지 학습에 성능이 좋은 Resnet-34에 대해 공부하여 실제 프로젝트 모델에 적용, 학습 및 평가 과정에서 얻은 손실과 정확도 등의 지표를 통해 확인과 모델의 예측 결과를 시각화하여 출력을 목표로 하였습니다.

2. 학습방법 및 구성

전이 학습(Transfer Learning)을 활용하여 이미지 분류 모델을 학습시키고 학습된 모델을 사용하여 주어진 한국 남자배우의 얼굴 이미지를 받고, 해당 배우의 식별 결과를 예측합니다.
Python을 위한 오픈소스 머신 러닝 라이브러리인 PyTorch와 torchvision 등의 라이브러리를 사용하였고, 단계에는 이미지 데이터 수집, 전처리, 모델 구성, 학습 과정, 평가 과정으로 구성하였습니다.

3. 적용 (향후 기술)

특정 배우와 관련된 광고 및 마케팅 캠페인을 개발할 수 있을 것이고, 사진 또는 동영상에서 한국 남자 배우의 얼굴을 식별하고 해당 배우에 대한 정보를 얻을 수 있는 App을 개발할 수 있을 것입니다.
또, 한국 남자배우 얼굴 구별뿐만 아니라 다른 이미지나 데이터들의 분류, 구별에 대한 연구나 응용에 도움을 줄 수 있을 것으로 기대되고 다양한 데이터 셋과 모델에 대한 실험과 성능 비교를 통해 더 정교한 이미지 분류 모델을 개발할 수 있을 것입니다.

4. 결론 및 느낀점

- 학습 및 평가 과정에서 얻은 손실과 정확도의 지표를 통해 확인
- 모델의 예측 결과를 시각화하여 출력
- 개인 프로젝트를 구상, 모델을 제작
- 프로젝트 구상, 설계에서 시간 소요
- 비록 시간이 많이 들긴 했지만 혼자서 데이터 셋을 구성하고 딥러닝 모델을 만들면서 개발자로서의 설계, 구성, 딥러닝의 역량을 키울 수 있는 프로젝트였다고 생각합니다.