딥러닝 기초 기말 Project

학번: 20215253

이름: 천애지

**# 서약**

아래 보고서는 **본인의 힘만으로 작성**해야 하며, 다른 학생에게 질문과 다른 학생의 코드를 참고 하는 행위는 모두 금지합니다

\* 수업에서 제공한 코드, 노트북은 모두 재활용 가능하며, 카피로 규정하지 않습니다

\* 수업 자료 이외에 참고자료가 있다면, 출처와 사용 부분에 모두 표시하는 경우는 모두 합당한 자료로 인정하겠습니다

\* 위에 대해서 모두 이해하고 동의했다면, 아래 `서약글`에 다음을 작성해주세요:

"본인은 위 서약글을 이해하고 동의하며, 프로젝트를 수행하는데 있어서 반칙을 할 경우 (제공자 포함) 본 프로젝트에 대한 점수가 반영되지 않는다는 것에 동의합니다."

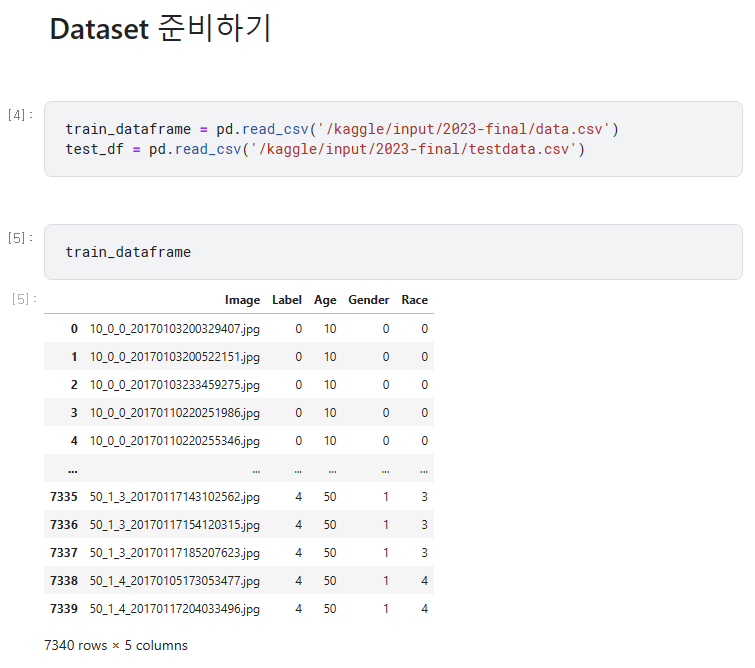
학번: 20215253

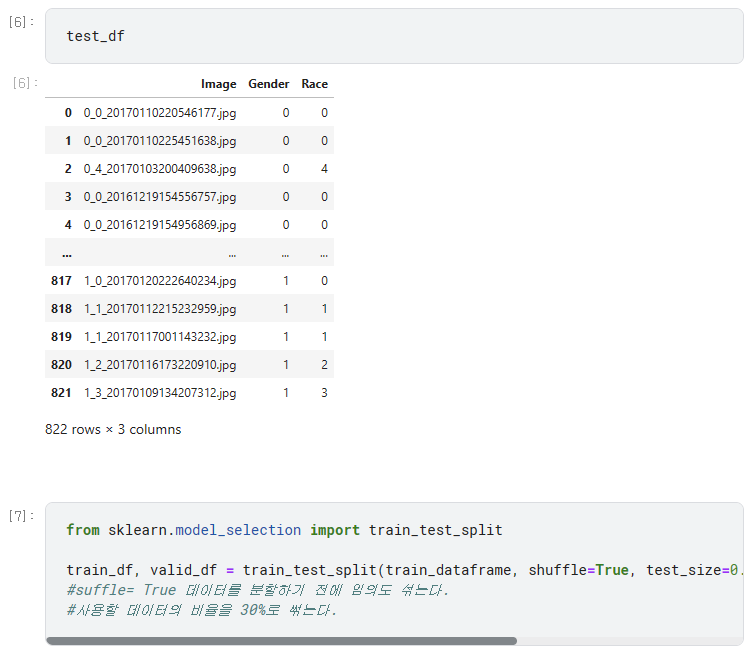
이름: 천애지

서약글: 본인은 위 서약글을 이해하고 동의하며, 프로젝트를 수행하는데 있어서 부정행위를 할 경우(제공자 포함)본 프로젝트에 대한 점수가 반영되지 않는다는 것에 동의합니다.

\*모든 코드에는 주석을 작성해 주세요

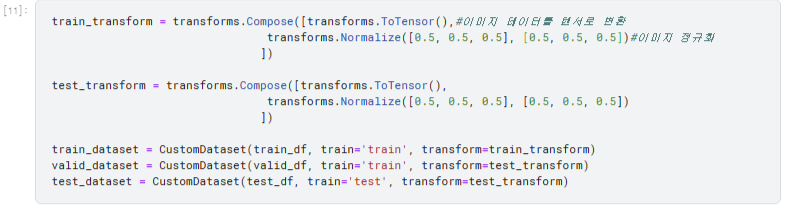
Step 1: Dataset 준비하기



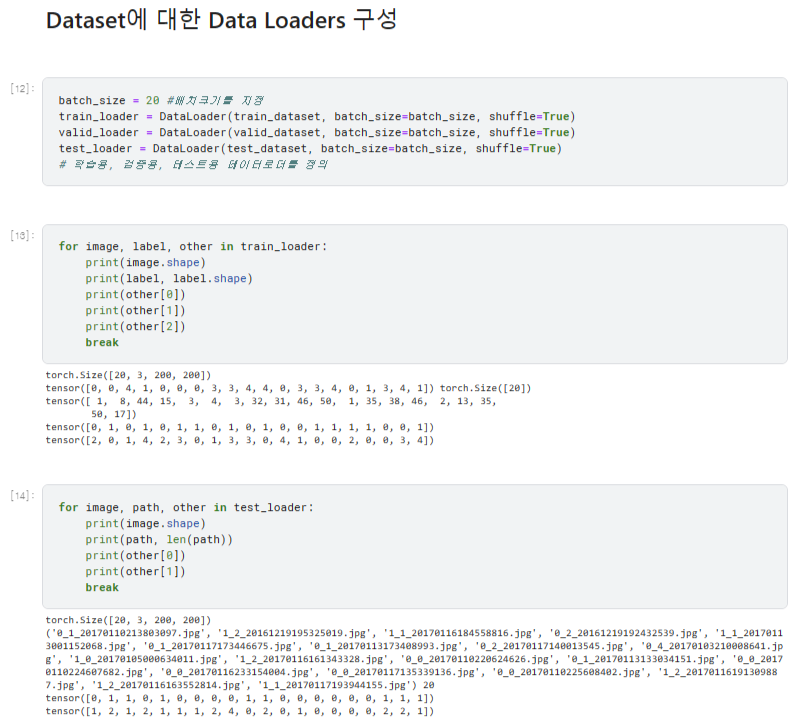








Step 2: Dataset 에 대한 Data Loaders 구성



데이터 전처리에 대한 설명

transforms.ToTensor()를 사용하여 이미지 데이터를 텐서(Tensor) 형태로 변환합니다.

또한, transforms.Normalize()를 사용하여 이미지를 정규화합니다. 정규화는 평균과 표준편차를 이용하여 데이터의 분포를 조정하는 과정입니다. 위의 코드에서는 [0.5, 0.5, 0.5]로 지정된 평균과 표준편차를 사용하여 이미지의 각 채널(RGB)을 동일한 범위로 조정합니다. 이를 통해 데이터의 분포를 중심으로 맞추고, 모델의 학습 안정성과 성능을 향상시킵니다.

원래 이미지를 랜덤으로 잘라 데이터를 늘리려고 했으나 데이터를 많이 늘려도 적중률이 올라가지 않아 사용하지 않았습니다.

Step 3: Neural Network 생성



설계한 모델을 출력 후 네트워크를 구성한 방법과 이유를 각 단계별로 설명

- 첫번째 convolutional layer(nn.Conv2d)는 입력 채널 수가 3(RGB 이미지)이고, 출력 채널 수가 16입니다. 입력 이미지의 각 위치에서 3x3 커널을 적용하여 특징 맵(feature map)을 생성합니다. 출력 특징 맵의 크기는 입력과 동일하게 유지되도록 패딩(padding)을 1로 설정했습니다.

- 두번째 convolutional layer은 입력 채널 수가 16이고, 출력 채널 수가 32고, 같은 방식으로 커널을 적용하고 패딩을 적용하여 특징 맵을 생성합니다.

- 세번째 convolutional layer은 입력 채널 수가 32이고, 출력 채널 수가 64입니다. 같은 방식으로 커널을 적용하고 패딩을 적용하여 특징 맵을 생성합니다.

- convolutional layer 다음에는 활성화 함수로 ReLU를 사용합니다. ReLU는 비선형성을 도입하여 모델이 복잡한 특징을 학습할 수 있도록 합니다.

- pooling layer (nn.MaxPool2d)는 특징 맵의 크기를 감소시키는 역할을 한다. 이 모델에서는 2x2 최대 풀링(max pooling)을 사용하여 특징 맵의 크기를 반으로 줄입니다.

- 특징 맵을 일렬로 펼치고(Flatten) 완전히 연결된 레이어(fully connected layer)에 전달하여 분류를 수행합니다.

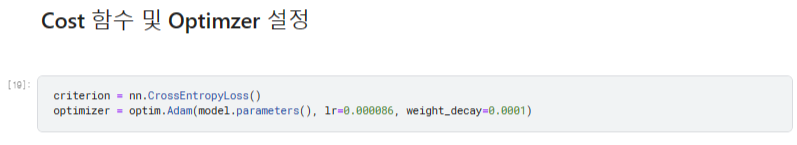
- 첫 번째 완전히 연결된 레이어(nn.Linear)는 입력 크기가 64x25x25(특징 맵의 크기)이고 출력 크기가 1100입니다. 이 레이어는 특징을 학습하고 복잡한 패턴을 추출하는 역할을 합니다.

- 두 번째 완전히 연결된 레이어는 입력 크기가 1100이고 출력 크기가 10입니다. 이 레이어는 최종적인 클래스 분류를 수행합니다.

- Dropout은 과적합(overfitting)을 줄이기 위해 사용되는 정규화(regularization) 기법입니다. 이 모델에서는 0.25의 드롭아웃 비율로 Dropout 레이어를 추가하여 학습 중에 뉴런의 연결을 임의로 끄는 것으로 일부 특징들의 공동 의존성을 줄이고 일반화 성능을 향상시킵니다.

이 모델의 장점이라고 생각하는 부분은 컨볼루션 레이어와 풀링 레이어를 통해 이미지의 공간적 구조를 캡처하고 특징을 추출할 수 있고, 완전히 연결된 레이어를 통해 추출된 특징을 기반으로 클래스를 분류할 수 있습니다. 또한 Dropout을 사용하여 과적합을 완화시킬 수 있습니다.

Step 4: Cost (Loss) Function 과 Optimizer 선택



Optimizer와 Cost 함수를 선정한 이유와 선정하는데 중요하다고 생각하는 내용을 모두 작성합니다.

1. Loss Function: - `nn.CrossEntropyLoss()`는 다중 클래스 분류 문제에 사용되는 손실 함수입니다. 이 함수는 모델의 출력과 정답 레이블 사이의 크로스 엔트로피 손실을 계산합니다. 모델의 출력은 클래스에 대한 로짓(logit) 값으로, `nn.CrossEntropyLoss()`는 로짓을 소프트맥스 함수를 통해 확률 값으로 변환하고, 실제 정답과의 손실을 계산합니다.]

2. Optimizer:

- `optim.Adam`은 아담(Adam) 최적화 알고리즘을 사용하는 옵티마이저입니다. 아담은 모멘텀(momentum)과 학습률 스케줄링(learning rate scheduling)을 자동으로 조절하면서 최적화를 수행하는 알고리즘으로, 일반적으로 딥 러닝 모델 학습에 효과적입니다.

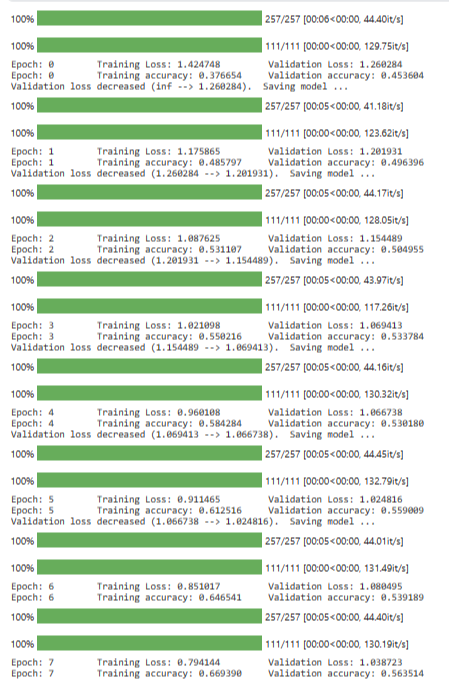
- `model.parameters()`는 모델의 학습 가능한 매개변수들을 가져옵니다. 이 매개변수들은 모델의 가중치(weight)와 편향(bias)입니다. `optim.Adam`은 이러한 매개변수들을 업데이트하기 위해 사용됩니다.

- `lr=0.000086`은 학습률(learning rate)로, 각 매개변수의 업데이트에 적용되는 스케일을 나타냅니다. 적절한 학습률을 선택하는 것은 모델의 학습 성능에 중요한 역할을 합니다.

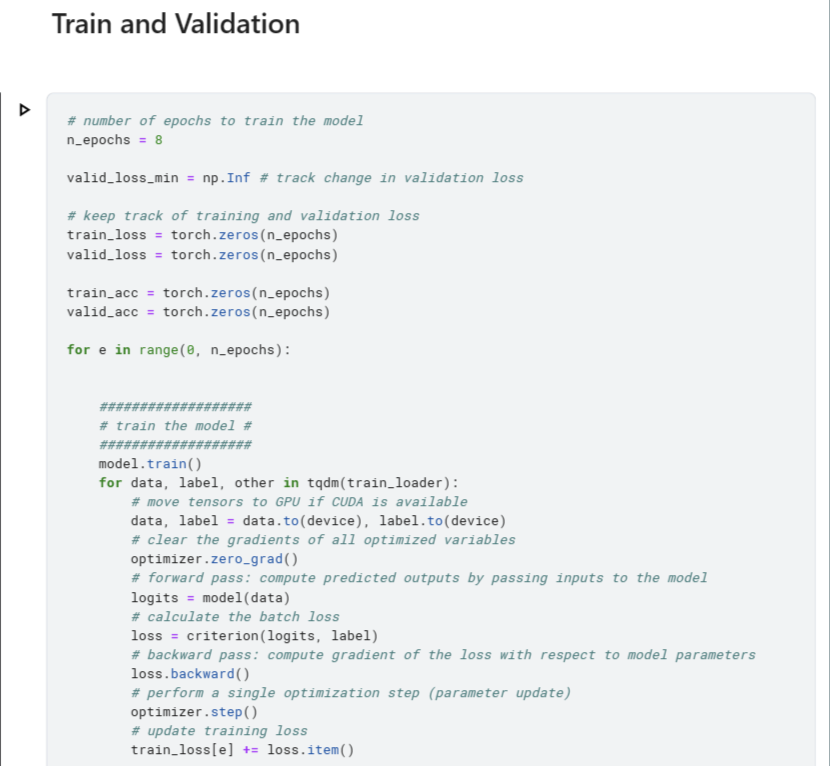
- `weight\_decay=0.0001`은 L2 정규화(L2 regularization)의 강도를 나타내는 값입니다. 정규화는 모델의 복잡도를 제어하여 과적합을 방지하고 일반화 성능을 향상시키는 데 도움을 줍니다.

- k-fold를 사용하려고 했으나 아직 딥러닝에 미숙하고 정확하게 알지 못하여 사용하지 않았습니다.

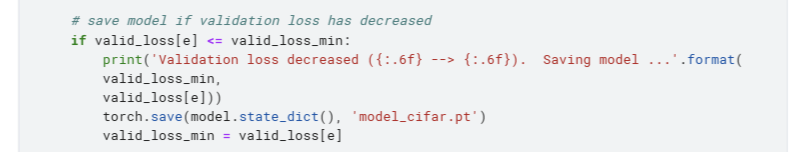
Step 5: 구성한 모델에 대한 Train and Validate 진행



Step 6: CNN model training/validation 분석

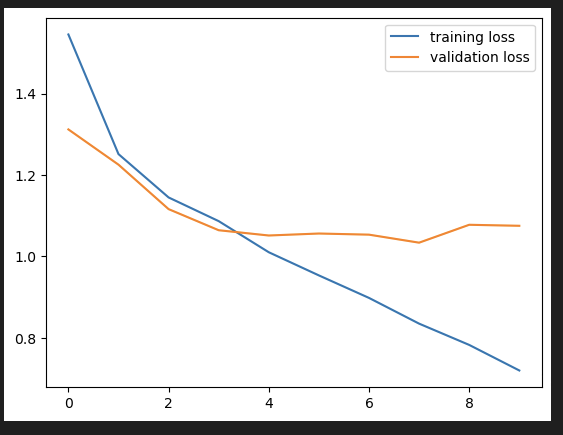


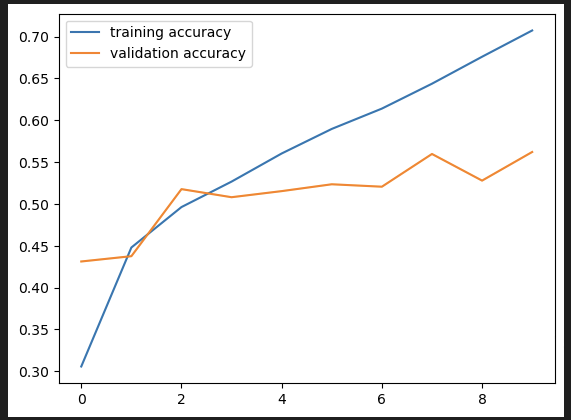




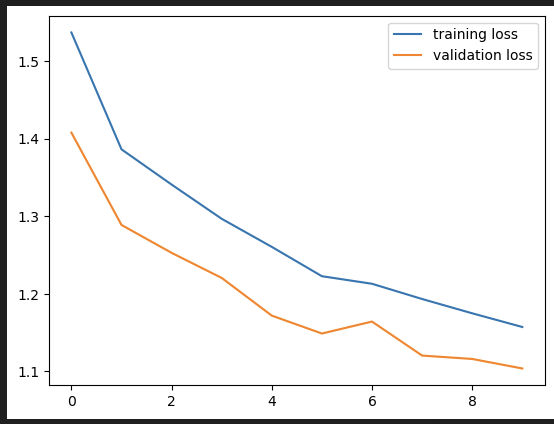
위에서 수행한 training + validation 과정을 설명하세요

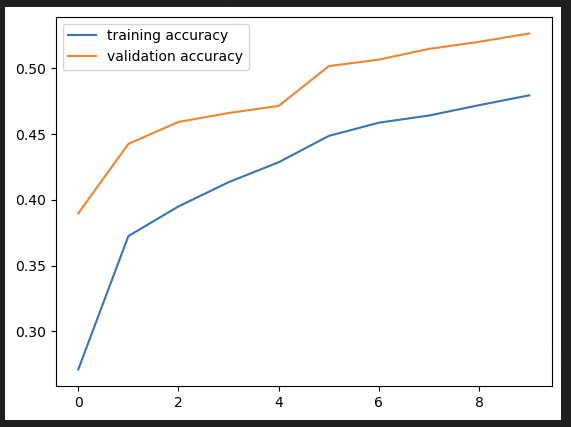
처음 lr을 너무 크게 잡아 overfitting이 발생하였는데 lr을 줄여가면서 overfitting이 발생하지 않도록 조절하였습니다.



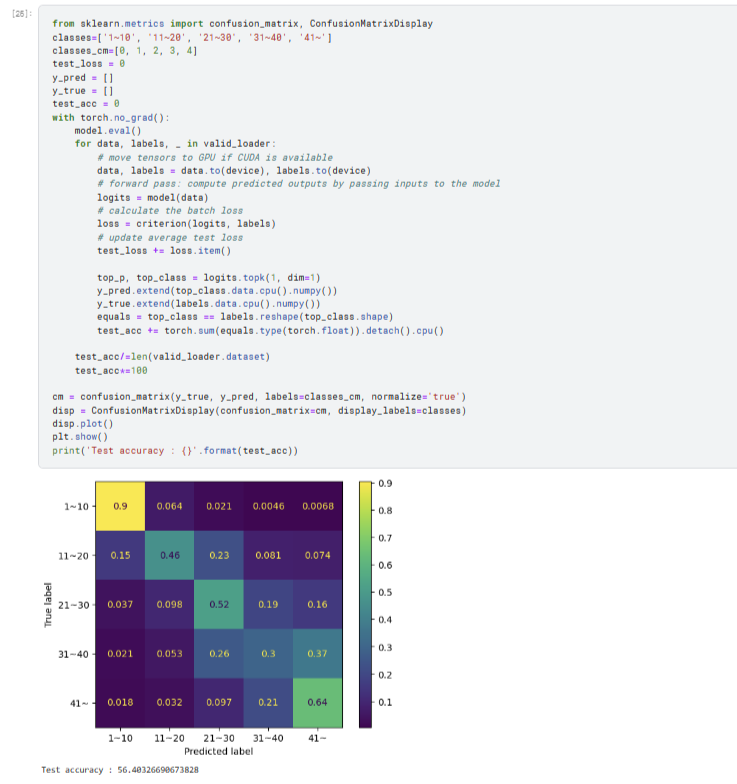


에서 lr을 조절하고 모델을 가볍게 만들어 조절하였습니다.





Step 7: Predict with Test Data



보완할점

- 적중률을 올리기 위해서 batch normalization, model ensembles, data augmentation등 하고 싶은 부분들이 많았지만 하는 방법을 몰라 하지 못했습니다 구글링으로 알아보기도 하였지만 코드만 보고는 이해할 수 없어서 방법들을 사용하지 않았습니다. 이러한 점이 너무 아쉽고 에폭 수를 올리면 학습률이 떨어질 때도 있고 에폭 수를 줄이면 또 학습률이 날 나올 때도 있어 이런 부분이 왜 그런지 알아보고 싶습니다.