딥러닝 기초 기말 Project

학번: 20195178

이름 : 서영재

**# 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: 20195178

이름: 서영재

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

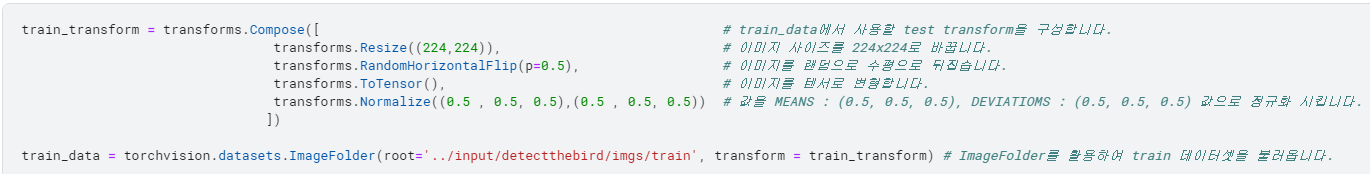
\*모든 코드에는 주석을 작성해 주세요

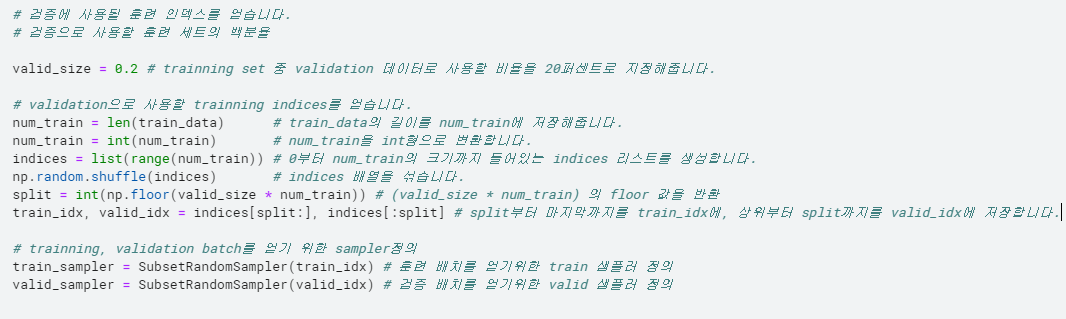
최종 제출시, 본 보고서와 .ipynb 노트북파일, test에 사용한 모델(.pt)파일을 압축해 제출해 주세요.

중요: 사용한 기법은 자신이 이해한 것 만을 사용하세요. 설명하지 않은 기법을 사용하면 그 부분을 제외하고 채점하겠습니다. 예를 들어서 자신의 힘으로 찾은 코드를 이용하려하는 경우 내용을 이해하고 보고서에 이해한 내용이 충분히 설명이 되어야만 사용을 허용합니다.

Step 1: Dataset 준비하기

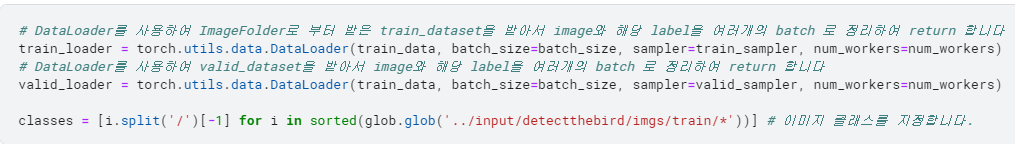
<코드 캡쳐 첨부>

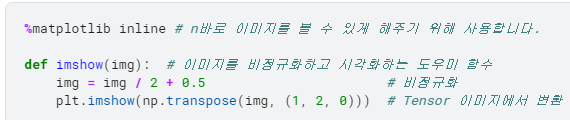




Step 2: Dataset 에 대한 Data Loaders 구성

<코드 캡쳐 첨부>







데이터 전처리에 대한 설명

위 작성한 코드에 대해서 설명하세요

• transforms.Resize(224,224) : 이미지 사이즈를 224x224로 변경하는 작업입니다.

• transforms.RandomHorizontalFlip(p=0.5) : 이미지를 랜덤으로 수평으로 뒤집는 작업입니다.

• transforms.ToTensor() : 이미지를 텐서로 변환시키는 작업입니다.

• transforms.Normalize((0.5 , 0.5, 0.5),(0.5 , 0.5, 0.5))

: (0.5, 0.5, 0.5), (0.5, 0.5, 0.5) 값으로 정규화된 torch로 변환시키는 작업입니다.

• torchvision.datasets.ImageFolder(root='경로', transform = transform)

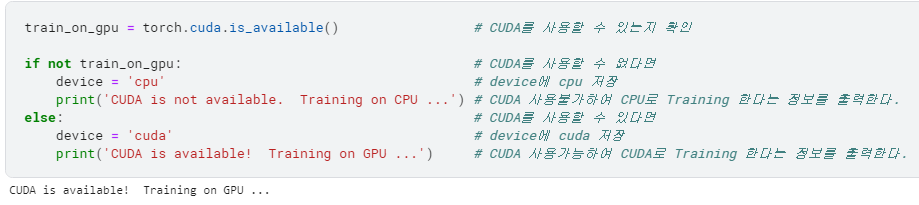
: ImageFolder를 활용하여 데이터셋을 불러옵니다.

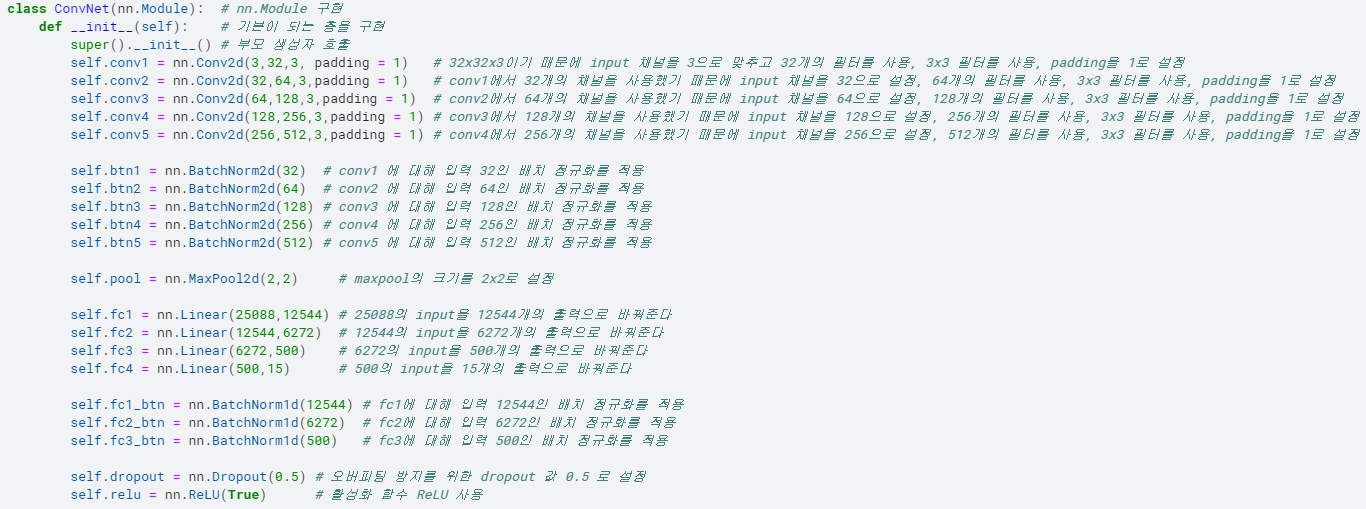
위 작업을 수행하여 학습에 필요한 데이터를 처리 가공하는 데이터 전처리를 수행하였습니다.

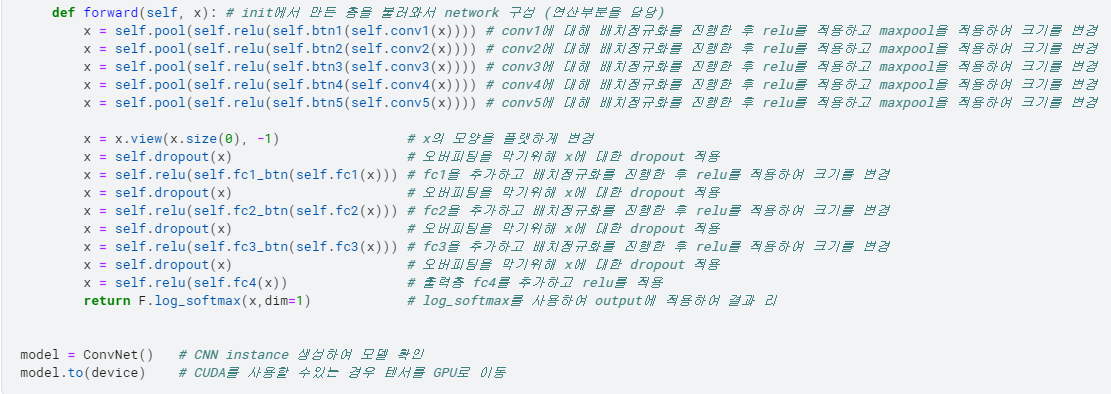
Step 3: Neural Network 생성

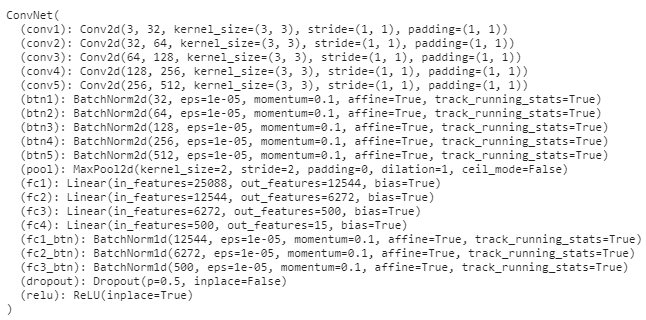
- Pretrained model을 허용하지 않습니다. (직접 모델을 설계해 주세요)

<코드 캡쳐 첨부>









설계한 모델을 출력 후 네트워크를 구성한 방법과 이유를 각 단계별로 설명

Ex) 차원 분석, 채널 수, 커널 크기, linear layer neuron 수

activation function과 선정 이유

최종 layer에서 사용한 activation function이 무엇인지 왜 사용하였는지

CNN 모델이 무엇을 입력 받고 출력하나요? 등…

<서술형>

5개의 convolution layer과 4개의 linear layer 구조를 가지고 있습니다.

* conv 1 : 3 x 32 , filter 수 32개 , padding = 1 , filter 크기: 3x3
* conv 2 : 32 x 64 , filter 수 64개, padding = 1 , filter 크기: 3x3
* conv 3 : 64 x 128 , filter 수 128개, padding = 1 , filter 크기: 3x3
* conv 4 : 128 x 256 , filter 수 256개, padding = 1 , filter 크기: 3x3
* conv 5 : 256 x 512 , filter 수 512개, padding = 1 , filter 크기: 3x3
* fc1 : 25088 x 12544
* fc2 : 12544 x 6272
* fc3 : 6272 x 500
* fc4 : 500 x 15 ( 네트워크의 최종 output layer로 output neuron의 수를 15로 설정 )

Activation function : ReLU

* 선정 이유 : 다른 activation function인 sigmoid, tanh함수보다 학습이 빠르고, 연산 비용이 적고, 구현이 간단하며, gradient가 출력층과 멀리 있는 layer 까지 전달된다는 성질로 인하여 데이터에 적합하도록 fitting이 잘되게 되고 이것으로 곡선 함수에 근사하도록 만들어 지게 되기 때문입니다.

최종 layer에서 사용한 activation function : LogSoftMax

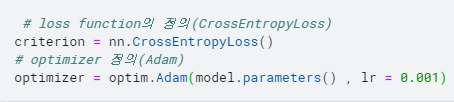
> 사용한 이유 : LogSoftMax는 softmax와 log함수의 결합한 함수로,

결과값이 포화되지 않고, 기울기 손실 문제를 피할 수 있기에 사용하였습니다.

CNN 모델은 train data을 입력 받아 해당 데이터에 대한 예측을 출력합니다.

Step 4: Cost (Loss) Function 과 Optimizer 선택

<코드 캡쳐 첨부>



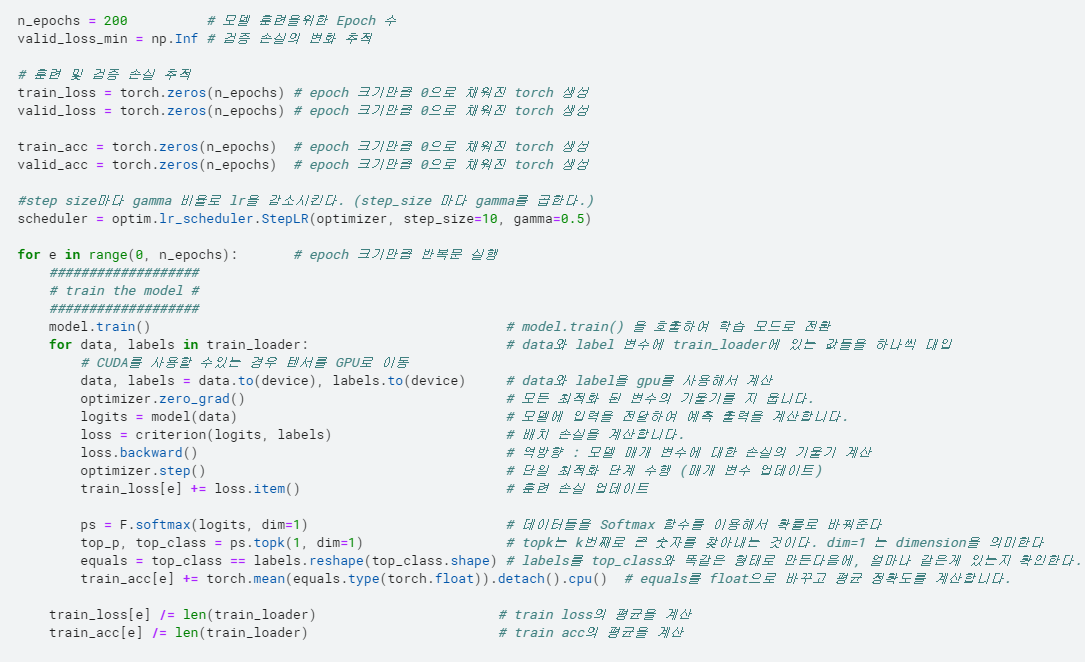
Optimizer와 Cost 함수를 선정한 이유와 선정하는데 중요하다고 생각하는 내용을 모두 작성합니다.

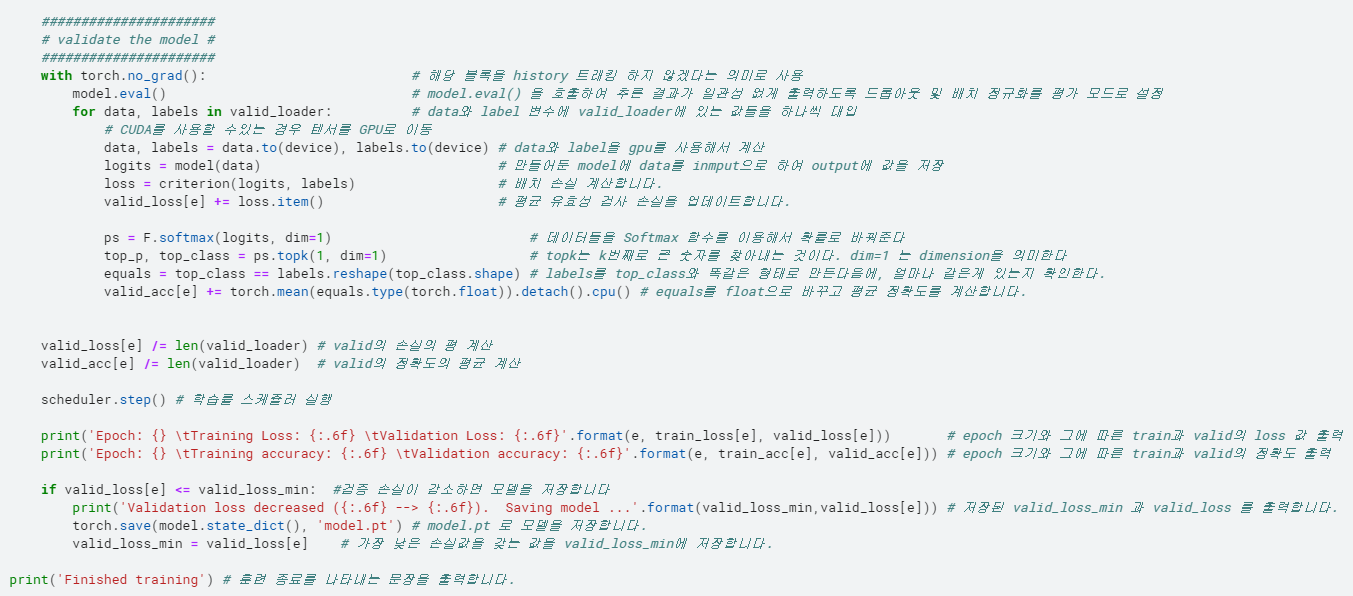
<서술형>

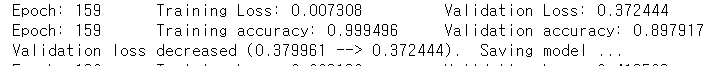
* Optimizer : Adam
* RMSprop와 모멘텀을 합친 방법으로, 방향과 학습률 모두를 잡기 위한 최적의 방법이라고 생각하였기에 선정하였습니다.
* Cost function이 단순한 것이 아닌 복잡하게 생긴 경우가 있기에 optimizer 함수가 필요합니다.
* Cost : CrossEntropyLoss
* 다중 분류에 사용되는 손실 함수로 연산을 한번에 처리하는 것이 수식이 간소화되어 역전파가 더 안정적으로 이루어지므로 선정하였습니다.
* 실제 데이터의 결과 값과 알고리즘을 통해 예측하는 값의 차이를 계산하여 Cost function을 최소화 시키는 알고리즘을 찾음으로써 궁극적으로 가장 적절한 예측을 할 수 있기에 중요합니다.

Step 5: 구성한 모델에 대한 Train and Validate 진행

<코드 캡쳐 첨부>



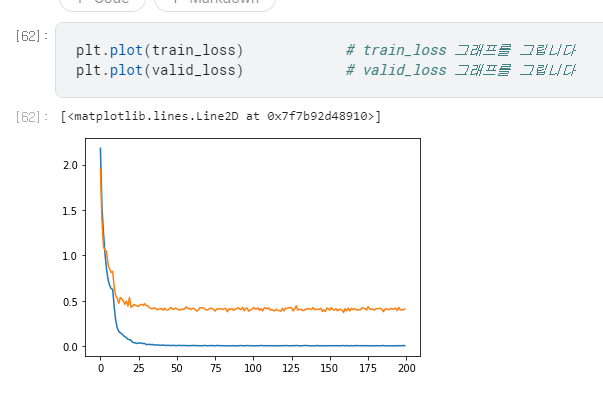


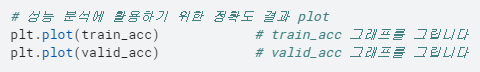


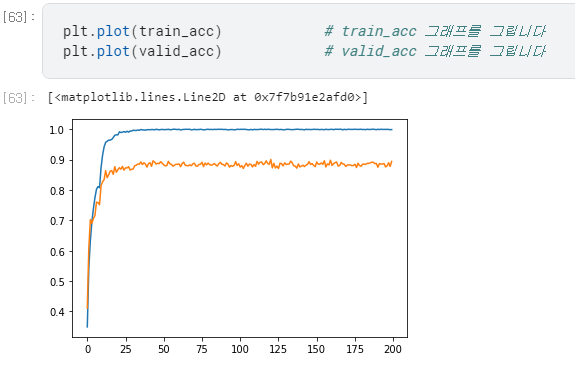
Step 6: CNN model training/validation 분석

<코드 캡쳐 첨부>











위에서 수행한 training + validation 과정을 설명하세요

training loss와 validation loss 그래프를 통해서 분석

Ex) hyper-parameter, model을 변경하면서 성능 개선한 과정을 최대한 설명하세요

overfitting, underfitting 분석 등..

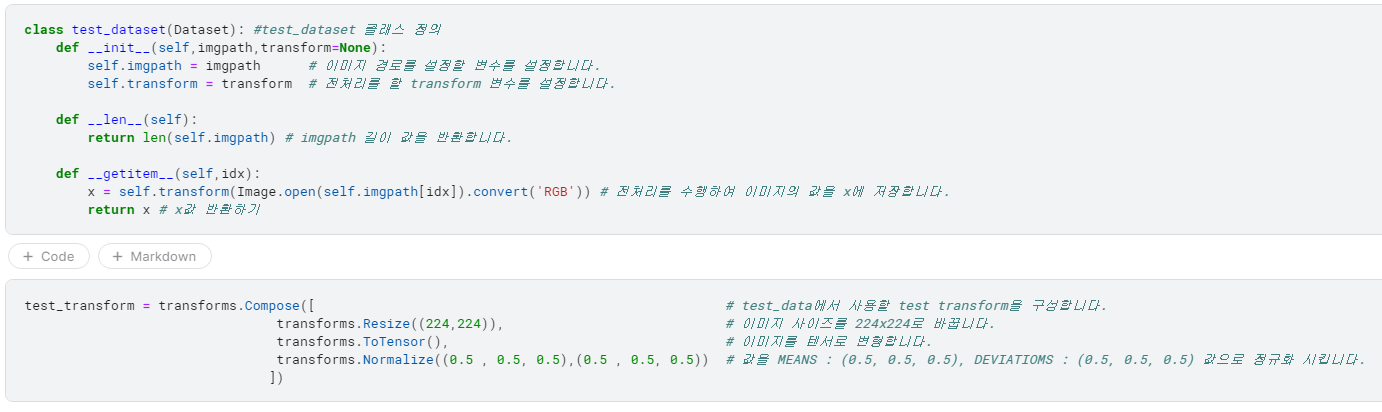
<서술형>

초반에 conv layer 3개, linear layer 2개로 이루어진 모델을 만들어 수행하니, 턱없이 부족한 결과가 나왔습니다. 8퍼센트라는 충격적인 정확도에 무엇이 문제인지 끊임없이 고민하였습니다. 그 결과 구조가 너무 얕다는 것을 알게 되었습니다. 이에 conv layer을 5개, linear layer 을 4개로 구성하고, 각 layer에 대해 배치 정규화와 relu함수를 적용하고, maxpooling을 시행하며 이전보다 높은 정확도를 얻을 수 있었습니다. 하지만, 그래프를 보며, overfitting 이 발생하고 있다는 사실을 알게 되었습니다. 이를 방지하기 위해, dropout을 과감히 0.8로 설정해보았더니, 정확도 60퍼센트의 결과를 얻을 수 있었습니다. 이후 epoch의 수도 50에서 100 그리고 200으로 설정하게 되어 더욱 좋은 결과를 얻을 수 있게 되었습니다. 최적의 optimizer 함수를 찾으려고 노력 끝에, Adam 이라는 함수에 learning rate를 0.001 로 두어 학습을 진행하였습니다.

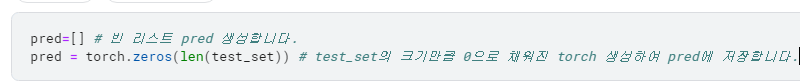
학습률 스케쥴러를 사용하여 학습률을 가변적으로 조정하여 model을 개선할 수 있었습니다.

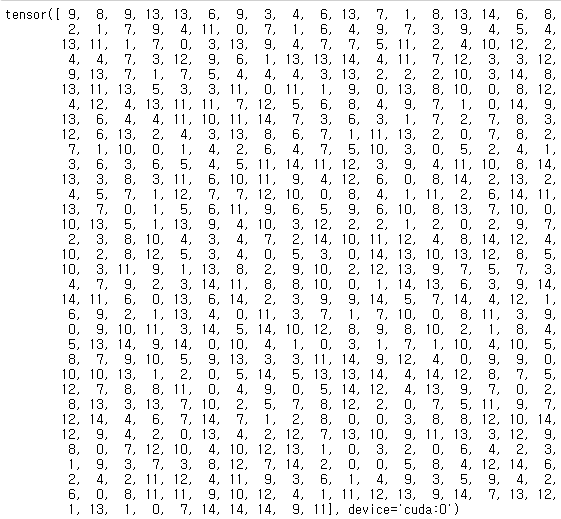
Step 7: Predict with Test Data

<코드 캡쳐 첨부>

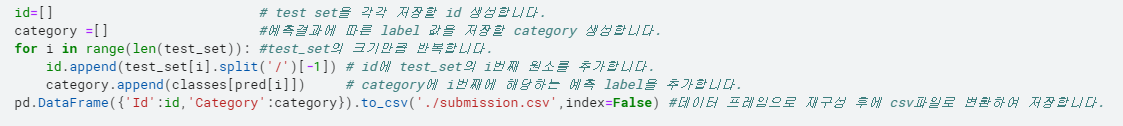












Step 8: Training Techniques

성능 개선을 위해서 사용한 기법 중에서 특별히 효과적이었던 부분이나 강조하고자 하는 내용을 작성해주세요.

Overfitting을 방지하기 위하여 Batch Normalization을 convolution층과 linear 각 층에 적용시켰습니다. 또한, dropout 값을 0.5로 설정하였습니다.

학습 향상을 위하여 학습률 스케쥴러를 사용하여 학습이 진행되면서 학습률을 그 상황에 맞게 가변적으로 적당하게 변경할 수 있었습니다. 그 결과 더 낮은 손실값을 얻을 수 있습니다. 학습률 스케쥴러 중 아래와 같이 지정한 스텝 단위로 학습률에 감마를 곱해 학습률을 감소시키는 방식의 스케쥴러를 사용하였습니다.

scheduler = optim.lr\_scheduler.StepLR(optimizer, step\_size=10, gamma=0.5)