**T3Q 플랫폼 Meta Pseudo Label[[1]](#footnote-1)**

**T3Q 플랫폼부문 기술연구소 인턴 김시연**

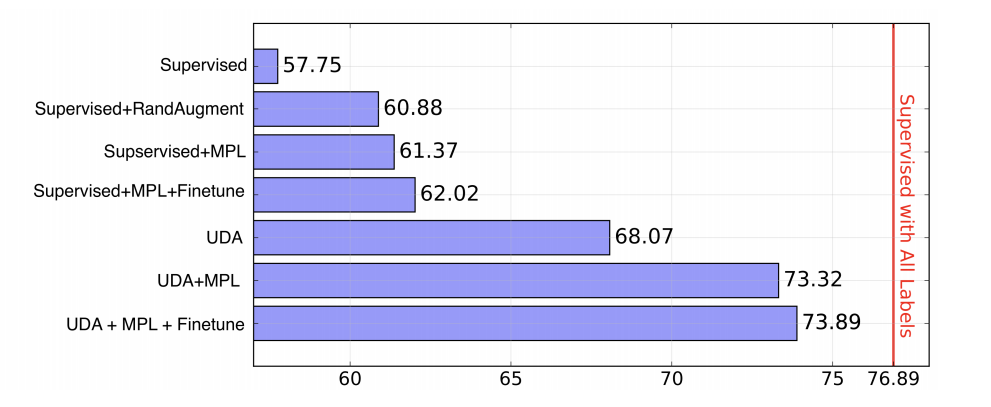
일반적인 deep learning model은 supervised learning model로 training data를 위한 labeling을 필요로 한다.하지만 사람이 직접 labeling을 다 진행하기에는 너무 많은 시간과 돈이 쓰인다. 고로 적은 양의 labeling을 통해 model을 training 시킬 수 있는 semi-supervised learning model을 사용하고자 했고 여러 semi-supervised learning algorithm 중 Meta Pseudo Labels이라는 알고리즘을 사용하게 되었다. 처음에는 MPL 모델을 online training 환경에서 돌아갈 수 있게 만들어보고 싶었으나 online training에 대한 이해가 부족해 우선적으로 MPL이라는 semi-supervised learning을 통해 model을 학습시킬 수 있는 코드를 플랫폼 위에 탑재하는 것으로 목표를 바꾸었다. MPL 코드는 github[[2]](#footnote-2)에 올라와있는 코드를 일부 수정하여 만들었다. Output은 MPL 알고리즘을 통해 training된 model.pt의 생성뿐만 아니라 추가적으로 최종 training 된 모델을 통해 unlabeled data에 labeling한 결과를 zip형태로 저장해보았다.

1. **Intro**
2. **Purpose**

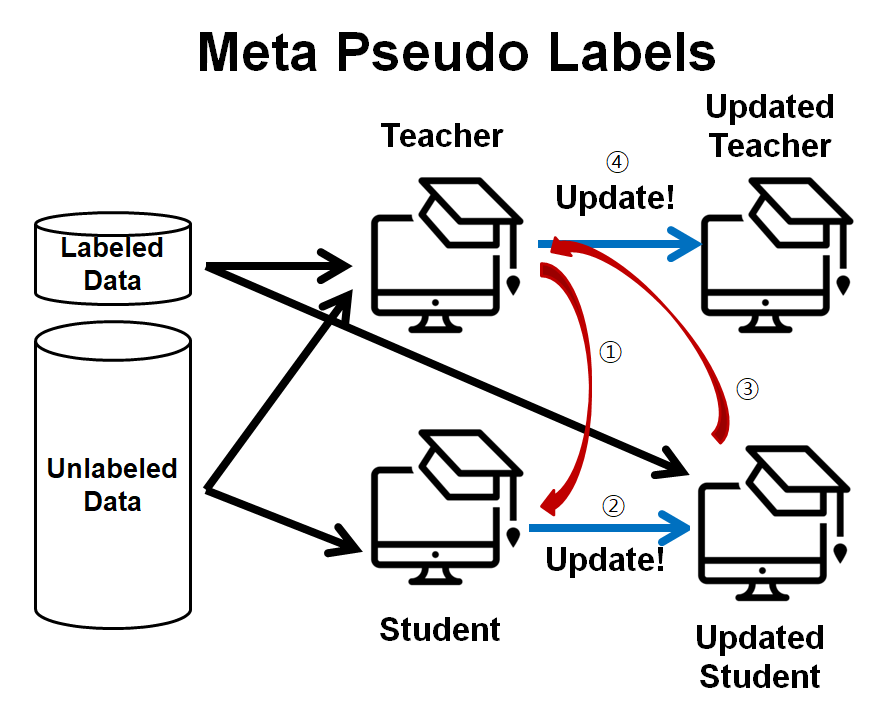
Meta Pseudo Labels라는 semi-supervised learning을 통해 unlabeled data가 존재할 때, 적은 양의 labeling만 진행해줘도 model을 학습시킬 수 있게 하고자 했다. 이로써 labeling에 소모되는 시간과 돈 등의 비용을 줄일 수 있다.

1. **Meta Pseudo Labels**

Meta Pseudo Labels 알고리즘은 teacher model과 student model, 총 2개의 학습 모델을 사용한다. 여기서 teacher이 학습하는 데에 UDA라는 unsupervised learning 기법이 사용된다.[[3]](#footnote-3) 이때 UDA를 사용하는 이유는 일반적으로 supervised learning만 사용하는 것 보다 성능이 훨씬 좋았기 때문이다.



이에 UDA 기법을 사용하기위해 augmentation이 우리가 사용할 Meta Pseudo Labels 코드에 쓰인다. 전체적인 MPL 과정을 설명하자면 다음과 같다.



크게 3가지 data가 존재한다. Labeled data, Unlabeled data, 그리고 augmented unlabeled data이다.

1. 먼저 teacher의 unlabeled data에 대한 pseudo label을 target으로 하여 student가 augmented unlabeled data의 predicted 결과와의 loss를 계산한다.
2. student가 계산한 loss로 student model이 updated된다.
3. updated되기 전 student model과 updated 된 student model 각각의 labeled data에 대한 predict 결과의 loss를 계산하고 그 차를 teacher의 loss 계산에 넘겨준다.
4. teacher의 loss는 크게 2가지 loss가 더해진다. t\_loss = t\_loss\_uda+t\_loss\_mpl. 여기서 t\_loss\_mpl를 계산하는 과정에서 ③번에서 계산된 student model의 labeled data에 대한 loss의 차가 사용된다. 그리고 t\_loss\_uda를 계산하는 과정에서 앞에서 설명한 UDA 알고리즘이 사용된다. 그리고 t\_loss로 teacher model을 update한다.
5. **전처리모듈**
6. **Composition**

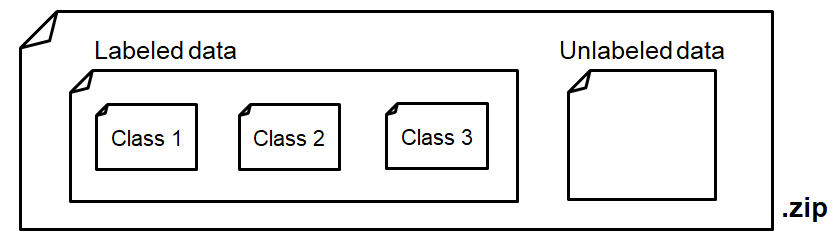
모듈명: mpl\_test

파일: run.py

역할: 데이터셋 압축 해제 코드



Input dataset: .zip 파일



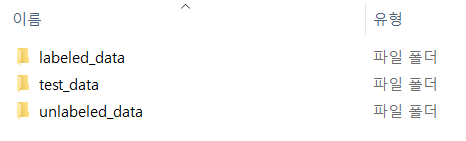
(추가적으로 test\_data가 존재하면 test\_data 폴더도 생성해야 한다. 그 형태는 labeled data 폴더와 유사하다)

1. **Input dataset 생성방법**

Ex) MNIST.zip

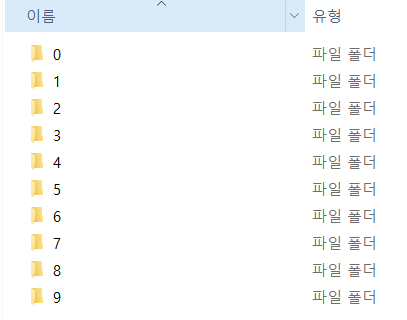
1. MNIST: labeled\_data, unlabeled\_data, (있다면 test\_data)로 폴더명을 정한다.

이때 dataset의 이름으로 폴더명을 정하도록 하자. Ex) MNIST

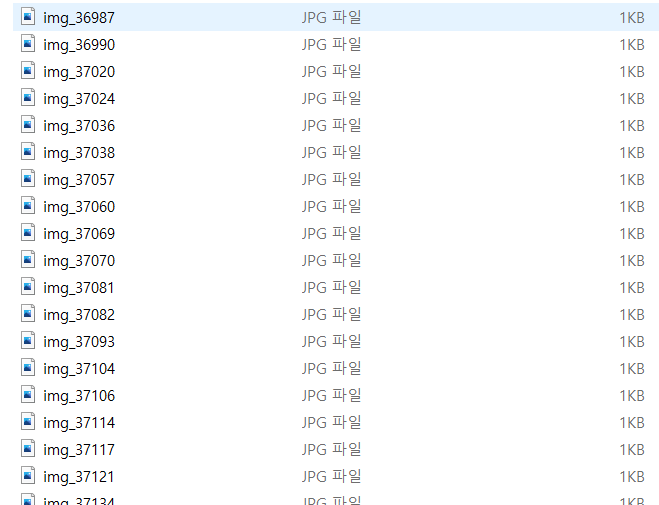


1. MNIST\labeled\_data: 각 이미지들의 label을 폴더이름으로 하여 이미지들을 넣어준다.

(있다면 MNIST\test\_data도 같은 형태로 구성한다.)



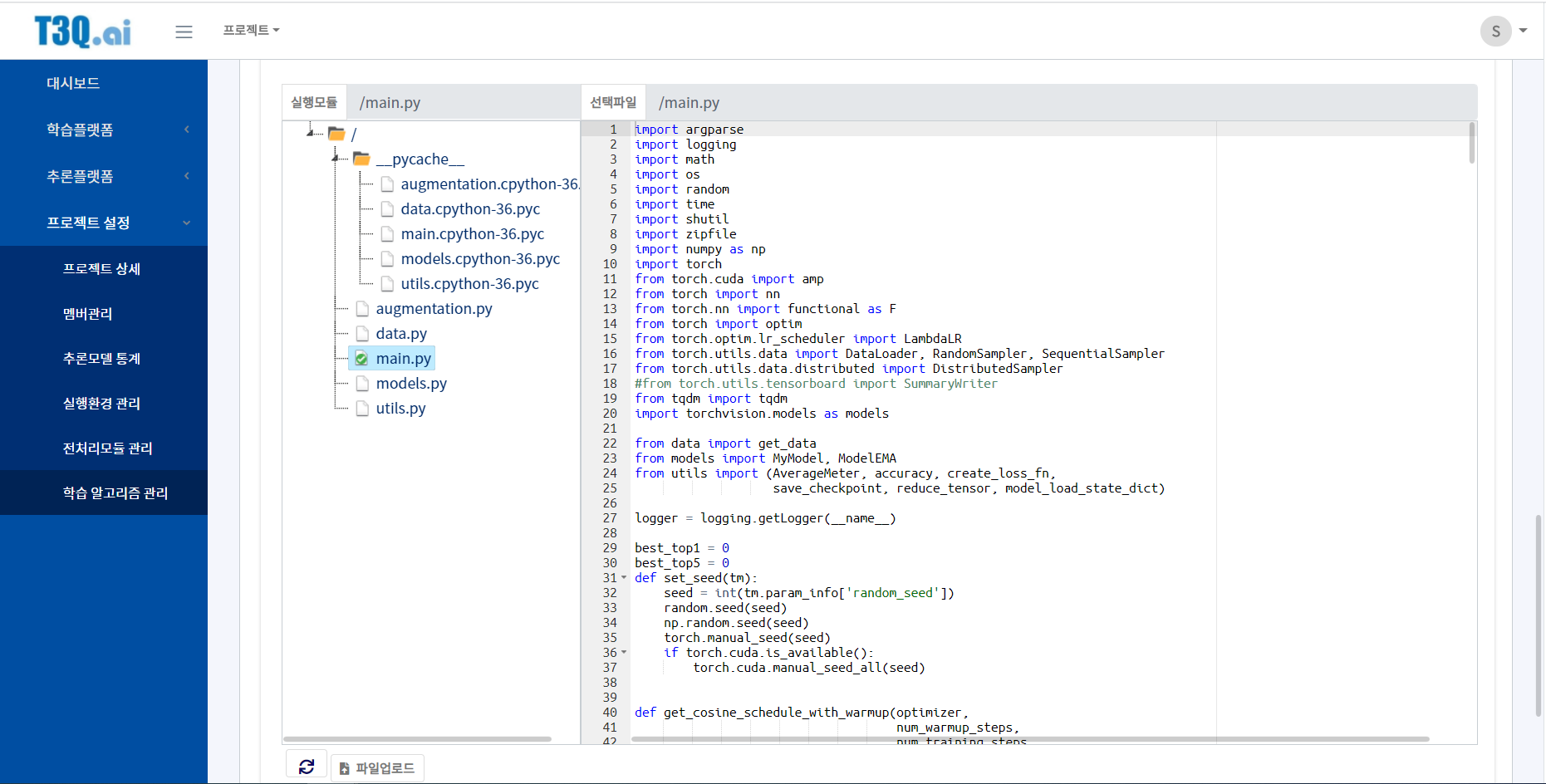
1. MNIST\labeled\_data\0,1,2,3,4,5,6,7,8,9, MNIST\unlabeled\_data 폴더 안에 각 label에 해당하는, 혹은 unlabeled된 image file을 넣어준다.



1. MNIST.zip으로 압축한다.
2. **학습 알고리즘**
3. **Composition**

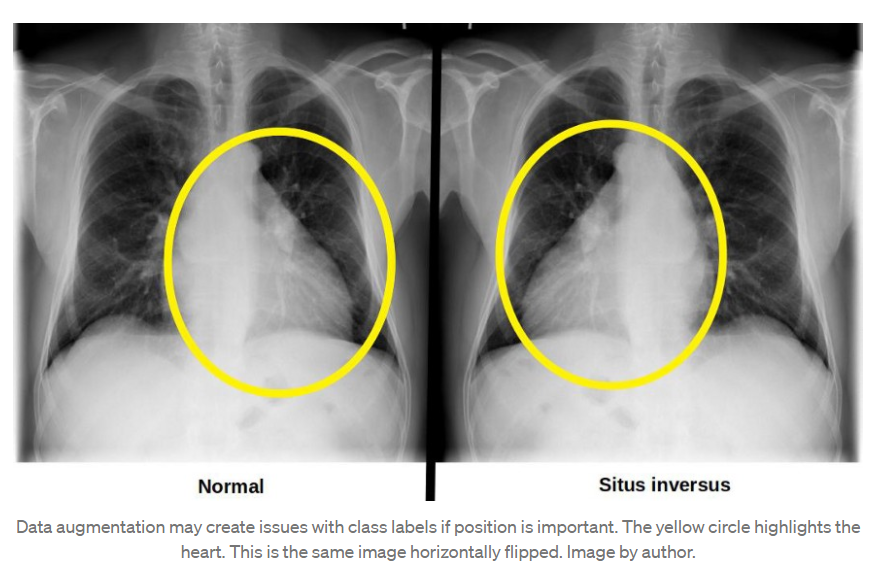
모듈명: mpl\_semifinal

파일: main.py (실행모듈), augmentation.py, data.py, models.py, utils.py



**Outline**

|  |  |
| --- | --- |
| 파일명 | 역할 |
| main.py(실행모듈) | MPL 알고리즘으로 train 및 evaluate, auto-labeling 진행 |
| augmentation.py | data에 가하는 augmentation 설정 |
| data.py | zip파일에서 dataset을 만들고 dataloader를 만든다. +augmentation을 가한다. |
| models.py | model EMA와 자기가 원하는 model이 있는 file. (EMA(exponential moving average)) |
| utils.py: | utils file |

* 데이터셋에 따라 augmentation.py, model.py에 수정이 필요할 수 있다. 이는 MPL 알고리즘과 관련이 있다. 우선 model.py의 경우, MPL은 알고리즘이지 training하는 network가 아니다. 고로 이를 사용자가 원하는 model를 정의해주어야 한다. augmentation의 경우 MPL 알고리즘 내에서 data의 augmentation을 진행해주는데, 이떄 데이터셋의 종류에 따라 training에 영향을 끼치는 augmentation이 있을 수 있으므로 이떄는 수동으로 주석처리를 해주는게 좋다. 아래와 같은 경우에는 좌우반전 augmentation이 training에 치명적인 영향을 끼칠 수 있다. 

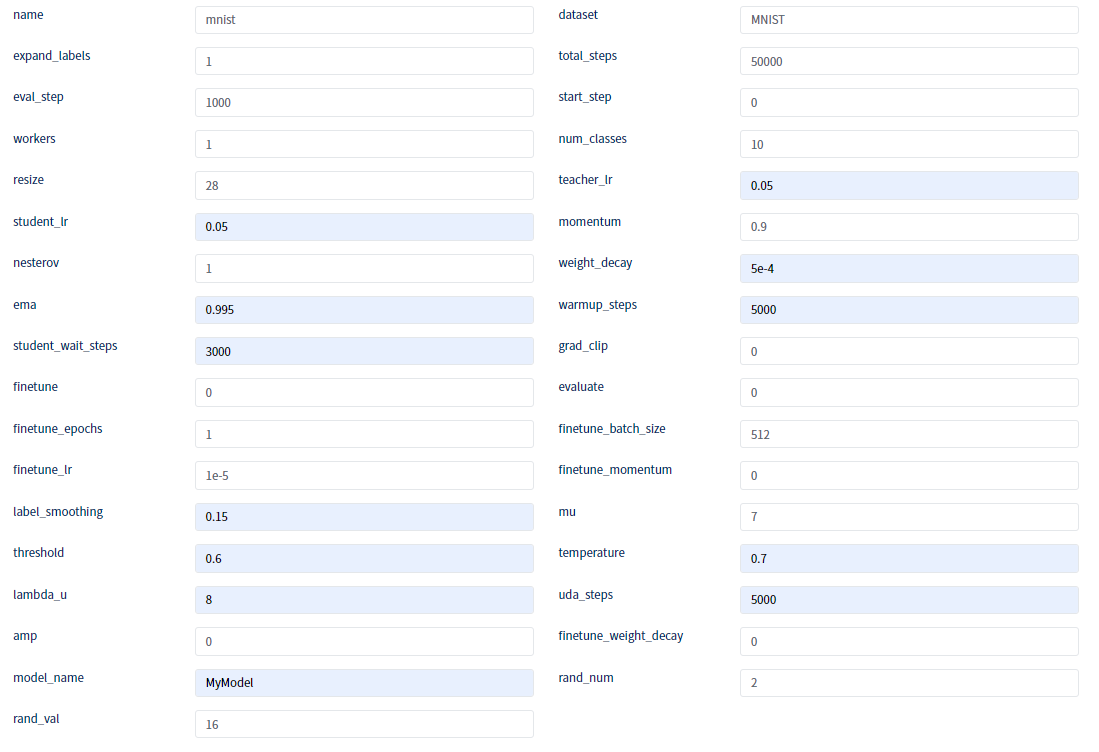
1. **parameter**

|  |  |  |
| --- | --- | --- |
| **About** | **parameter** | **설명** |
| **DataLoader** | dataset | dataset의 이름 (뒤에서 더 자세히 다룰예정) |
| batch\_size |  |
| workers |  |
| **Optimizer** | teacher\_lr, student\_lr |  |
| momemtum: | Determining how much to reflect the previous gradient |
| nesterov |  |
| weight\_decay | large-> underfitting, small-> overfitting |
| **Scheduler** | warmup\_steps |  |
| total\_steps |  |
| student\_wait\_steps (student Scheduler) |  |
| **GPU** | amp\_val | amp.autocase, amp.GradScaler: related to 16-bits training |
| **Augmentation** | rand\_num | transformation 종류 수 |
| rand\_val | transformation 정도 value |
| **그밖에** | seed | random seed |
| mu | coefficient of unlabeled loss (this is involved in unlabeled dataloader batchsize) |
| ema | EMA decay rate |
| lambda\_u | coefficient of unlabeled loss |
| grad\_clip | gradient norm clipping |
| threshold | pseudo label threshold |
| temperature | pseudo label temperature : as this goes small, become similar to one hot encoding |
| \* resume | If your training stopped, start from checkpoint |

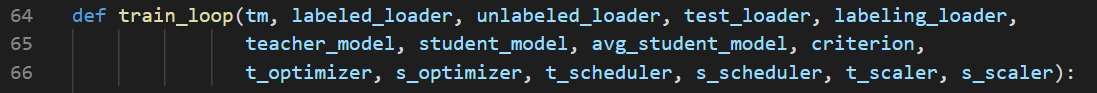
그밖에 ‘name’ parameter는 log를 통한 실험명 확인으로 크게 중요한 요소가 아니다.

\*\*batch\_size는 labeled data의 파일 수보다 작아야하며, batch\_size\*mu는 unlabeled data의 파일 수보다 작아야한다.

ex) parameter 설정예시



1. **main.py**



: MPL 알고리즘으로 training



: evaluation, test\_data가 있는 경우에는 test\_data로 evaluation을 진행하고 없는 경우에는 labeled data로 진행한다. 후자의 경우 accuracy가 당연히 전자보다 높게 나온다.



: fine-tuning

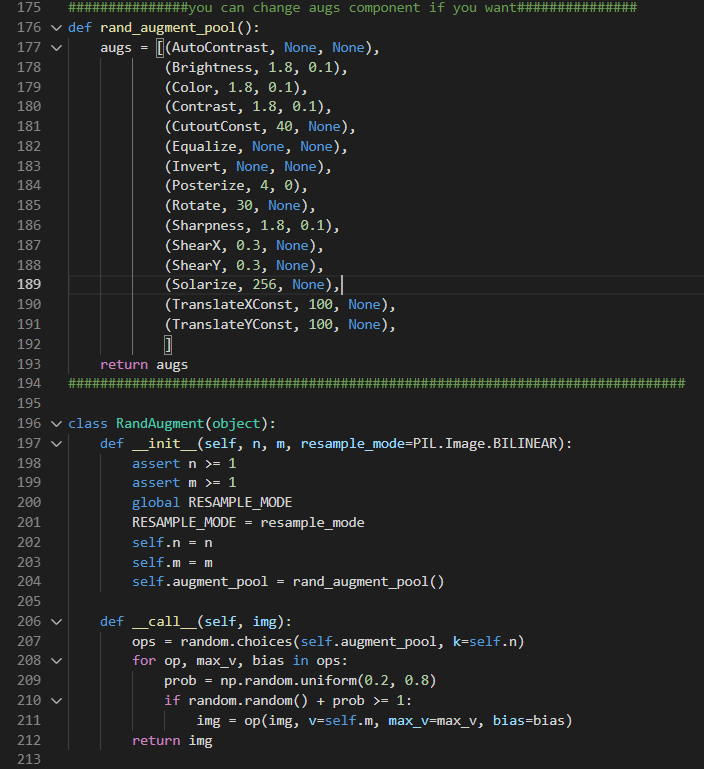


: unlabeled dataset에 training된 model로 labeling을 진행하여 zip파일로 tm.model\_path에 저장한다.



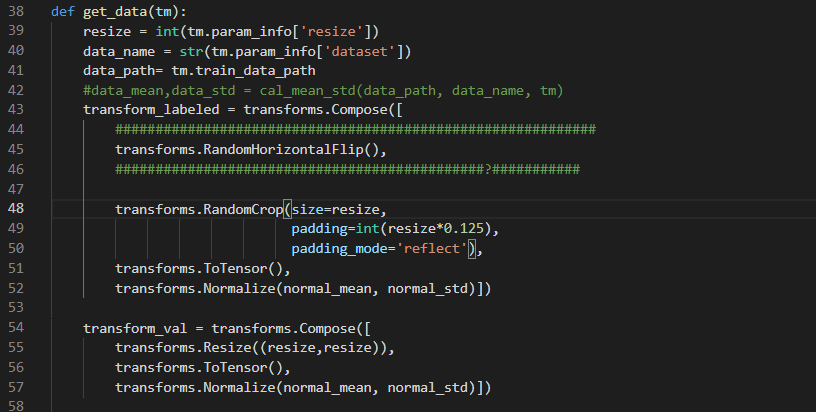
: dataloader, teacher model, student model, criterion, optimizer 등을 선언한다.

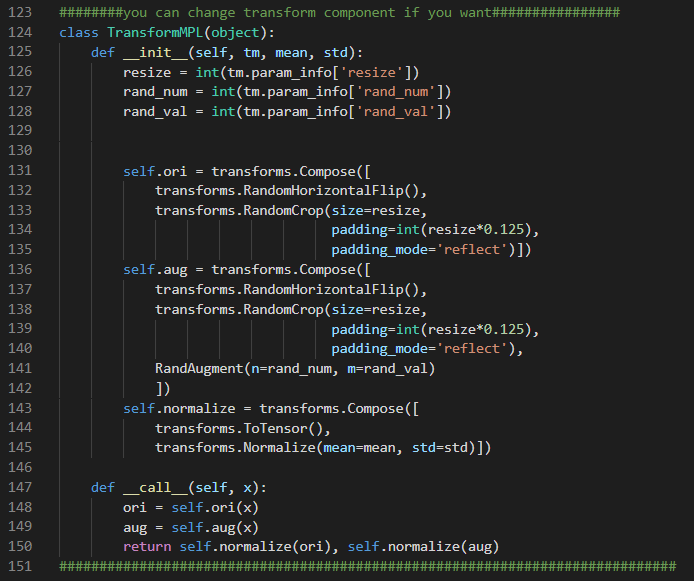
1. **augmentation.py**



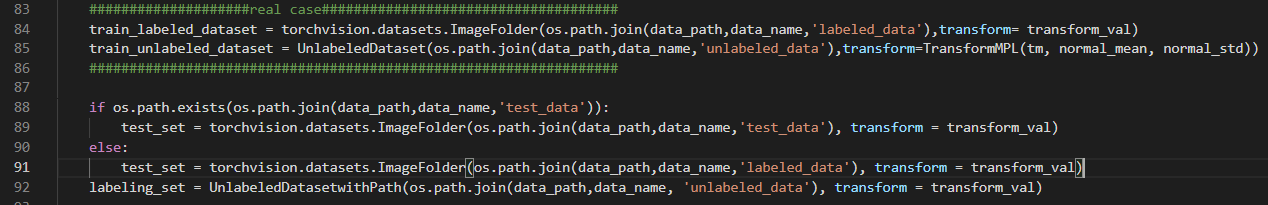
위의 outline에서 언급한 것처럼 불필요한 augmentation은 rand\_augment\_pool() 함수의 augs안의 요소들에 주석처리를 하면 된다. 물론 추가하는 것도 가능하다.

1. **data.py**
2. 우선 augmentation에서와 같은 이유로 아래의 두 부분에 있어 데이터셋에 따라 수정이 필요하다.





1. 앞에서 input data를 만드는 법에 대해 상세히 기제했는데, 이는 data.py에서 다음과 같은 경로로 dataset을 받아오기 때문이다.



여기서 data\_name에 dataset 폴더의 이름이 들어간다. 이는 플랫폼 parameter ‘dataset’에서 기입해준다.

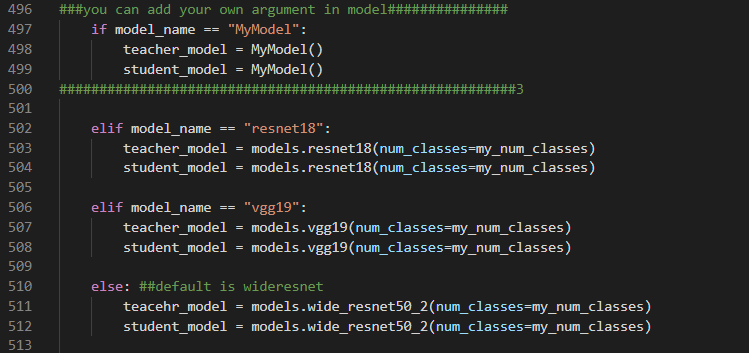
1. **models.py**
2. ModelEMA

ModelEMA는 student 모델의 parameter(weight)를 받아 EMA(exponential moving average) 연산이 이루어진다. 플랫폼에서 'ema’라는 이름의 parameter를 통해 ema 계산 유무를 선택할 수 있다.

1. MyModel



class MyModel에 자신이 원하는 network를 만들고 플랫폼에서 'model’ parameter에서 'MyModel’을 입력한다. 참고로 main.py에서 다음과 같이 을 통한 여러 내장된 network들을 지원한다.

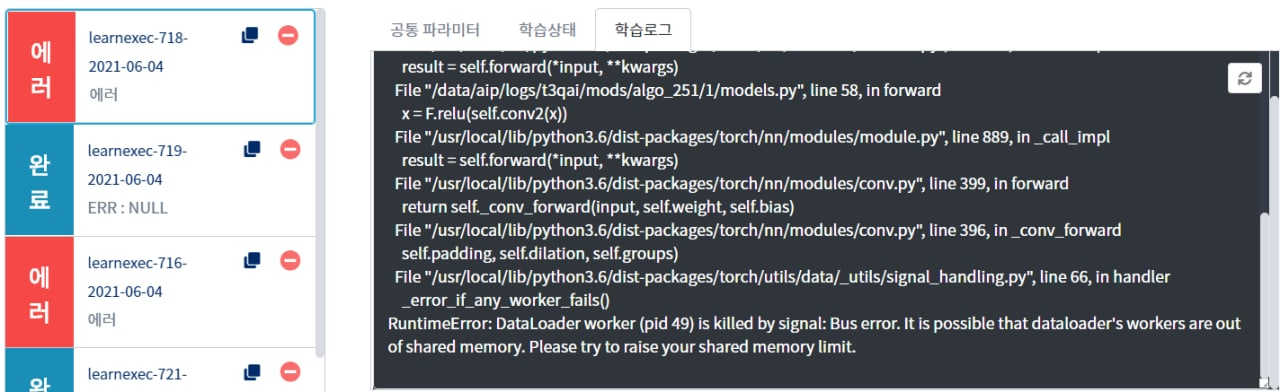


1. **Issue**

① tm.model\_path에 접근하는 법이 확실하지 않아서 resume에 대한 argument는 일단 빼두었다. checkpoint를 저장은 하지만 확인해보지는 못한 상태이다.

② batch\_size가 labeled data의 수보다 작아야 하며, batch\_size\*mu가 unlabeled data의 수보다 작아야 한다 -> GPU memory와 image data의 수를 잘 고려하여 batch\_size와 mu를 결정해야한다.

③ resize의 크기가 너무 크면 memory error가 뜬다.



1. https://arxiv.org/pdf/2003.10580.pdf [↑](#footnote-ref-1)
2. https://github.com/kekmodel/MPL-pytorch [↑](#footnote-ref-2)
3. https://arxiv.org/pdf/1904.12848.pdf [↑](#footnote-ref-3)