1. 알약 분류 유효성 검증 준비

알약 분류 기능의 유효성을 검증하기 위해서는 아래와 같이 사전에 필요한 구성요소들이 필요하다.

1. 실행에 필요한 python package 설치.
2. 폴더로 구분되어진 알약 이미지들
3. 실행에 필요한 file 들

그리고 기본적으로 실행환경은 Ubuntu 18.04, Python3.8 을 기준으로 한다.
(다른 버전으로 해도 크게 문제 되지 않는다.)

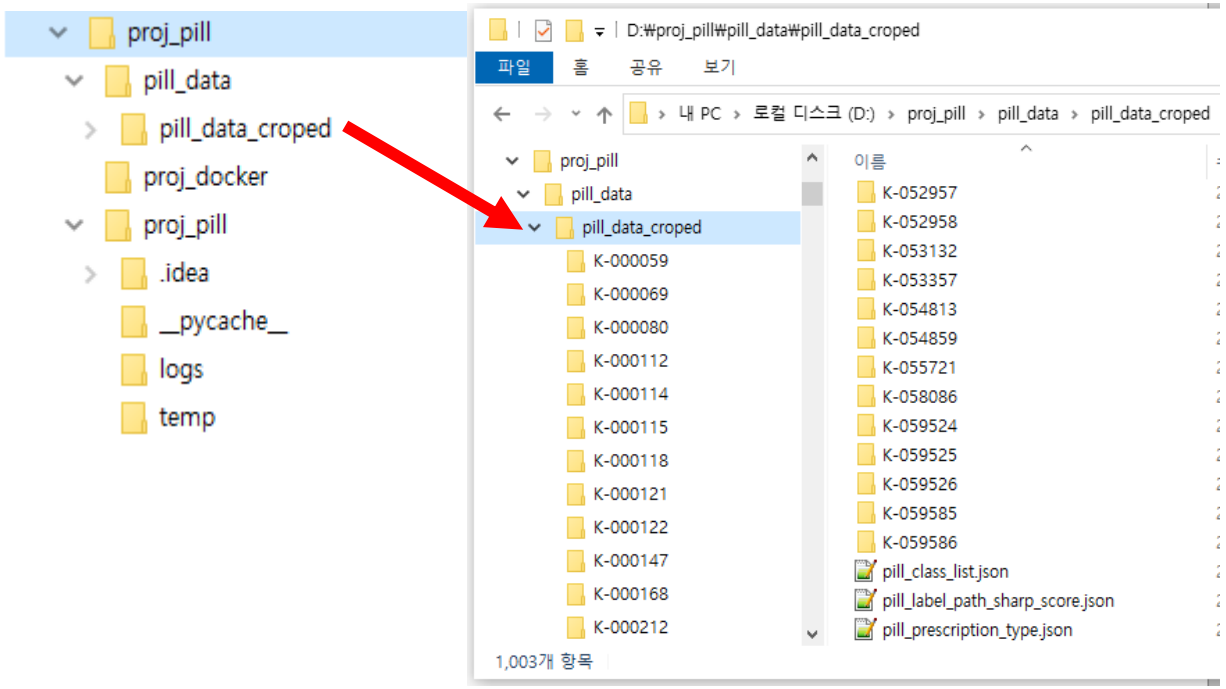
▪ 실행에 필요한 python package 설치

Ubuntu 18.04와 Python 3.8이상이 설치 된 것을 가정하고, 아래 package을 설치한다.

- Nvidia cuda driver 10.2 (혹은 11.0 이상)설치
- pip install torch torchvision (https://pytorch.org/ 참조)
- pip install numpy
- pip install opencv-python
- pip install imgaug
- pip install PIL
- pip install tqdm
- pip install codecs
- pip install json
- pip install matplotlib

▪ 폴더로 구분되어진 알약 이미지들

알약을 구분할 수 있는 ID이름으로 폴더를 구성하고 그 안에 해당 알약 이미지들이 존재하도록 구성한다.



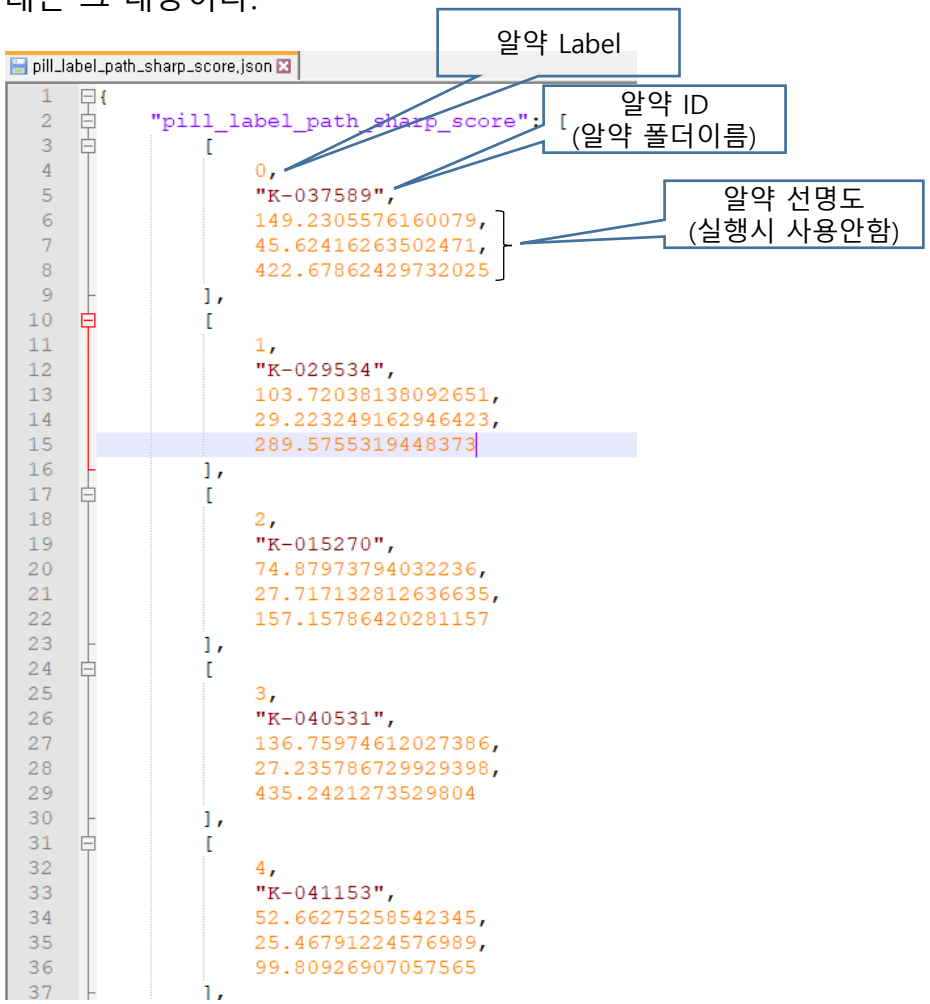
위 그림처럼 proj_pill/pill_data/pill_data_croped 안에 알약 folder을 구성한다.

추가적으로 제공하는 pill_label_path_sharp_score.json과 pill_class_list.json 이 그림과 같이 위치해 있어야 한다.

1. 알약 분류 유효성 검증 준비

pill_label_path_sharp_score.json 의 목적은 알약에 대한 ID와 Label을 정의하고 실행시 참조하게 되어 있다.

아래는 그 내용이다.



pill_label_path_sharp_score.json

```
{
    "pill_label_path_sharp_score": [
        [
            0,
            "K-037589",
            149.2305576160079,
            45.62416263502471,
            422.67862429732025
        ],
        [
            1,
            "K-029534",
            103.72038138092651,
            29.223249162946423,
            289.5755319448373
        ],
        [
            2,
            "K-015270",
            74.87973794032236,
            27.717132812636635,
            157.15786420281157
        ],
        [
            3,
            "K-040531",
            136.75974612027386,
            27.235786729929398,
            435.2421273529804
        ],
        [
            4,
            "K-041153",
            52.66275258542345,
            25.46791224576989,
            99.80926907057565
        ],
```

pill_class_list.json 의 목적은 알약의 class와 용도을 정의하고 실행시 참조하게 되어 있다.

Class0 : 카메라 각도가 90도, 75도로 캡쳐된 알약이미지의 list
Class1 : 카메라 각도가 70도, 60도로 캡쳐된 알약이미지의 list

```
{
"pngfile_class0_train": [
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-026788/K-026788_0_0_0_2_90_240_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-046428/K-046428_0_2_0_2_75_340_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-001438/K-001438_0_0_1_0_75_140_200.png",
...
"pngfile_class0_valid": [
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-046428/K-046428_0_2_0_1_90_140_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-046428/K-046428_0_1_0_0_90_220_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-001728/K-001728_0_2_1_0_90_020_200.png",
...
"pngfile_class0_test": [
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-052619/K-052619_0_2_0_2_75_140_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-006250/K-006250_0_1_0_1_75_000_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-038576/K-038576_0_0_0_2_75_120_200.png",
...
"pngfile_class1_train": [
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-044085/K-044085_0_0_1_1_70_340_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-009458/K-009458_0_2_0_0_60_240_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-023720/K-023720_0_1_1_2_60_060_200.png",
...
"pngfile_class1_valid": [
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-024752/K-024752_0_2_1_2_60_320_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-037589/K-037589_0_2_0_2_70_140_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-037589/K-037589_0_2_1_2_60_180_200.png",
...
"pngfile_class1_train": [
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-037043/K-037043_0_2_0_1_60_080_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-018254/K-018254_0_2_1_0_70_200_200.png",
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-005949/K-005949_0_0_1_0_60_260_200.png",
...
}
```

위의 내용에서
'pngfile_class0_train' 은 class0에 해당하고, 모델 학습용임을 의미함.
'pngfile_class0_valid' 은 class0에 해당하고, 모델 검증용임을 의미함.

1. 알약 분류 유효성 검증 준비

또한 위의 내용에서
'pngfile_class0_test' 은 class0에 해당하고, 모델 시험용임을 의미함.
'pngfile_class1_train' 은 class1에 해당하고, 모델 학습용임을 의미함.
'pngfile_class1_valid' 은 class1에 해당하고, 모델 검증용임을 의미함.
'pngfile_class1_test' 은 class1에 해당하고, 모델 시험용임을 의미함.

그리고 이미지 파일이 절대경로이므로, 이 경로를 맞추어야 함.
"/home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/K-026788/K-026788_0_0_0_2_90_240_200.png",

** 절대경로를 python file에서 바꿀 수 있지만, 편의상 고정경로만 설명함.
** 학습용, 검증용, 시험용 폴더는 존재하지 않음. '**pill_class_list.json'** file이 기능을 대신함.

- **실행에 필요한 file 들**

- 실행할 python file들은 이미 학습한 모델 파일이 필요한다. 아래와 같다.

pill_resnet152_dataclass0_aug0.pt : 알약 class0으로 학습한 모델 파일
pill_resnet152_dataclass1_aug0.pt : 알약 class1으로 학습한 모델 파일
pill_resnet152_dataclass01_aug0.pt : 알약 class0, class1으로 학습한 모델 파일

- 실행할 python file은 아래와 같다.

main_cls0.py : 알약 class0을 시험할 실행 파일.
main_cls1.py : 알약 class1을 시험할 실행 파일.
main_cls01.py : 알약 class0, class1을 시험할 실행 파일.

- 그 외 필요한 파일들.
get_cli_args.py : 환경변수을 만들거나, 수정할 수 있다.
gen_pill.py : 이미지들을 읽고, 학습할 수 있게 dataset을 만든다.
pill_classifier.py : 학습과 검증을 시행하는 code.
make_label_sharpness.py : data 준비용
make_pill_class_list.py : data 준비용.

2. 알약 분류 유효성 검증 시행

2.1 알약 분류 검증 전 아래와 같이 환경 폴더가 지정되어 있다고 가정한다.

~/proj/proj_pill/proj_pill : python 파일과 모델 파일이 위치한 경로.

~/proj/proj_pill/pill_data/pill_data_cropped :
→ 이미지가 폴더로 구분되어 위치한 경로
→ pill_class_list.json, pill_label_path_sharp_score.json 가 존재함.

2.2 알약 class0에 대해 아래와 같이 실행한다.

```
(base) root@55e94ecbb291:/# cd /home/ubuntu/proj/proj_pill/proj_pill
(base) root@55e94ecbb291:/home/ubuntu/proj/proj_pill/proj_pill# python main_cls0.py
job=resnet152 run_phase:test aug_level:0, dataclass:0
BATCH_SIZE:8, num_workers:4, num_threads:2
model_path_in is /home/ubuntu/proj/proj_pill/proj_pill/pill_resnet152_dataclass0_aug0.pt
dataset dir is /home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/pill_class_list.json
run_phase is test, aug_level is 0
gen_type is read_only_image, loading data ...
label_path was loaded from   <<< pngfile_class0_test >>>
data loading done.  dataset'length is 64740
valid dataset was loaded
dataset loading time is 3.2785534858703613
optimizer was selected as type:sgd
model_path will be loaded from:/home/ubuntu/proj/proj_pill/proj_pill/pill_resnet152_dataclass0_aug0.pt
model was loaded from state
Mon Dec  6 03:08:52 2021
resnet152 aug_level:0 :test Epoch  #0: 100%|████████████████████████████████████████████████████████████████████████████████| 8093/8093 [58:21<00:00,  2.31it/s, loss=0.0111, top1=99.7, top5=100]
Epoch: [0][8092/8093]   Data time 3501.450 (3501.450 Now:2021-12-06 04:07:14.246878)    Loss 718.4540 (0.0111)Accuracy top1:99.7312, top5:100.0000
Mon Dec  6 04:07:14 2021
job done
(base) root@55e94ecbb291:/home/ubuntu/proj/proj_pill/proj_pill#
```

→위의 실행은 모델명:reset152, Batch size:8, 모델파일:pill_resnet152_dataclass0_aug0.pt, dataset길이:64740, optimizer:sgd
→ top1:99.7, top5:100

2. 알약 분류 유효성 검증 시행

2.3 특정한 알약 이미지를 모델이 예측한 label을 확인하려면
실행폴더에서 'dir_testimage' sub 폴더를 만들고, 확인하려는 알약이미지를 넣어둔다.

그리고 아래 처럼 실행한다.

```
(horovod) ubuntu@gpu-1:~/proj/proj_pill/proj_pill$ python main_cls01_dir.py
job=resnet152 run_phase:test aug_level:0, dataclass:01
BATCH_SIZE:64, num_workers:4, num_threads:2
model_path_in is /home/ubuntu/proj/proj_pill/proj_pill/pill_resnet152_dataclass01_aug0.pt
valid dataset was loaded
2021-12-09 15:14:48.593789: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.10.1
model was loaded from state
Thu Dec  9 15:14:58 2021
resnet152 aug_level:0 :test Epoch  #0:   0%|                                                    | 0/1 [00:00<?, ?it/s]
/home/ubuntu/anaconda3/envs/horovod/lib/python3.9/site-packages/torch/nn/functional.py:718: UserWarning: Named tensors and all their associated APIs are an experimental feature and subject to change. Pl
ease do not use them for anything important until they are released as stable. (Triggered internally at  /pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
resnet152 aug_level:0 :test Epoch  #0: 100%|████████████████████████████████████████████| 1/1 [00:01<00:00,  1.06s/it, loss=29.4, top1=0, top5=0]
Epoch: [0][0/1] Data time 1.064 (1.064 Now:2021-12-09 15:14:59.822028)  Loss 58.7487 (29.3743)Accuracy top1:0.0000, top5:0.0000
Thu Dec  9 15:14:59 2021
('K-045583_0_0_0_0_60_340_200.png', 'K-051669_0_0_0_0_60_100_200.png')
[743, 494]
job done
(horovod) ubuntu@gpu-1:~/proj/proj_pill/proj_pill$ █
```

→ 위의 실행은 'K-045583_0_0_0_0_60_340_200.png', 'K-051669_0_0_0_0_60_100_200.png' 이미지가 각각 [743, 494] 으로 예측되었음을 표시한다.

2. 알약 분류 유효성 검증 시행(Docker이용시)

1. Docker 설치하기

```
~$> sudo apt-get update

~$> sudo apt-get install  apt-transport-https ca-certificates curl gnupg-
agent  software-properties-common

~$> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
~$> sudo add-apt-repository  "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs)   stable"

~$> sudo apt-get update
~$> sudo apt-get install docker-ce docker-ce-cli
```

설치 확인.

```
~$> sudo docker version
```

2. 알약용 image을 Docker에 load 하기

```
ubuntu@gpu-1:~$ cd proj/proj_pill/proj_docker/

ubuntu@gpu-1:~/proj/proj_pill/proj_docker$ dir
total 4964700
drwxrwxr-x 2 ubuntu ubuntu       4096 Nov 15 15:27 .
drwxrwxr-x 5 ubuntu ubuntu       4096 Nov 15 15:20 ..
-rw-rw-r-- 1 ubuntu ubuntu 5083836928 Nov 15 15:07 pill_class.tar


ubuntu@gpu-1:~/proj/proj_pill/proj_docker$ sudo docker load -i
pill_class.tar
824bf068fd3d: Loading layer
[==================================================>]  65.51MB/65.51MB
0677e35507df: Loading layer
[==================================================>]  3.576GB/3.576GB
dce55ae465d9: Loading layer
[==================================================>]  1.326GB/1.326GB
cc1427064650: Loading layer
[==================================================>]  116.3MB/116.3MB
Loaded image: ubuntu:pill

ubuntu@gpu-1:~/proj/proj_pill/proj_docker$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
ubuntu       pill      8d50ddab7868   45 minutes ago   4.96GB

ubuntu@gpu-1:~/proj/proj_pill/proj_docker$
```

2. 알약 분류 유효성 검증 시행(Docker이용시)

3. 알약용 docker image을 실행하기

```
~$> ubuntu@gpu-1:~/proj/proj_pill/proj_docker$ sudo docker run -it -v /home/ubuntu/proj/proj_pill:/home/ubuntu/proj/proj_pill ubuntu:pill

(base) root@7cce986fb0b5:/#
```

Docker 내부의 작업위치

Docker와 연결된 외부 directory
외부 director에 python file과 시험할 이미지가 있음.

4. 알약 class 0 에 대해 시험해보기

```
(base) root@7cce986fb0b5:/# cd /home/ubuntu/proj/proj_pill/proj_pill

(base) root@7cce986fb0b5:/home/ubuntu/proj/proj_pill/proj_pill# python main_cls0.py
job=resnet152 run_phase:test aug_level:0, dataclass:0
BATCH_SIZE:8, num_workers:4, num_threads:2
model_path_in is /home/ubuntu/proj/proj_pill/proj_pill/pill_resnet152_dataclass0_aug0.pt
dataset dir is /home/ubuntu/proj/proj_pill/pill_data/pill_data_croped/pill_class_list.json
run_phase is test, aug_level is 0
gen_type is read_only_image, loading data ...
label_path was loaded from   <<< pngfile_class0_test >>>
data loading done.  dataset'length is 64740
valid dataset was loaded
dataset loading time is 1.5846672058105469
optimizer was selected as type:sgd
model_path will be loaded from:/home/ubuntu/proj/proj_pill/proj_pill/pill_resnet152_dataclass0_aug0.pt
model was loaded from state
resnet152 aug_level:0 :test Epoch  #0:   0%|                                  | 14/8093 [00:48<7:43:26,  3.44s/it, loss=0.00693, top1=100, top5=100]
```

알약 class 1 에 대해 estimate 하려면,
'python main_cls1.py'

주의: 1. 사용되는 docker image은 GPU을 지원하지 않음.
2. docker은 시험환경만 제공하고, 내부에 source code와 이미지를 포함하지 않는다(외부 연결 사용).

3. 실행파일 설명

main_cls0.py

```
from pill_classifier import *
from get_cli_args import get_cli_args


job = 'resnet152'
if __name__ == '__main__':
    # job = 'hrnet_w64'
    job = 'resnet152'
    args = get_cli_args(job=job, run_phase='test', aug_level=0, dataclass='0')

    print(f'model_path_in is {args.model_path_in}')

    end = time.time()
    if args.run_phase == 'train'  :
        args.dataset_train = Dataset_Pill(args, args.json_pill_class_list,  transform=transform_normalize, run_phase='train')
        print(f'train dataset was loaded')

    args.dataset_valid = Dataset_Pill(args, args.json_pill_class_list, transform=transform_normalize, run_phase='test' if args.run_phase == 'test' else 'valid')
    print(f'valid dataset was loaded')


    print(f'dataset loading time is {time.time() - end}')

    pill_classifier(args)
    print('job done')
```

train: 학습시
valid: 검증시
test : 시험시

수행에 필요한 알약정보를 가져옴.

수행할 함수를 call함.
Pill_classifier.py에 있음

3. 실행파일 설명

pill_classifier.py

```
from get_cli_args import get_cli_args
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms, models
import torch.optim as optim
import torch.backends.cudnn as cudnn
from torch.optim.lr_scheduler import ReduceLROnPlateau

from gen_pill import Gen_Digit
from hrnet import get_hrnet
from utils import model_load, model_save, accuracy, get_optimizer, transform_normalize, AverageMeter
import time
from tqdm import tqdm
from torch.utils.tensorboard import SummaryWriter
import os
import datetime
```

```
class Dataset_Pill(Dataset):
    def __init__(self, args, dir_dataset, transform=None, target_transform=None, run_phase='train'):
        self.args = args
        self.gen_digit = Gen_Digit(args, dir_dataset, run_phase)
        self.transform = transform
        self.target_transform = target_transform
        self.run_phase = run_phase

    def __len__(self):
        return self.gen_digit.len_total

    def __getitem__(self, idx):
        image, label, path_img, aug_name = self.gen_digit.generate_digits_by_index(self.args, idx)

        if self.transform is not None:
            image = self.transform(image)

        if self.target_transform is not None:
            label = self.target_transform(label)
        if self.run_phase == 'valid' or self.run_phase == 'test':
            return image, label, path_img, aug_name
        else:
            return image, label

def get_pill_model(args):

    if args.cnn_name == 'resnet152' :
        model = models.resnet152(num_classes=args.num_classes)
    elif args.cnn_name == 'hrnet_w64' :
        model = get_hrnet()
        model.classifier = nn.Linear(in_features=2048, out_features=args.num_classes, bias=True)
    else:
        raise Exception('No Found CNN Name')

    if args.cuda == True:
        if args.gpu is not None:
            model.cuda(args.gpu)
        else:
            model.cuda()
    else:
        model.cpu()

    return model
```

pill_classifier.py

```python
def train(args, dataloader, sampler, model, criterion, optimizer, epoch, log_writer=None, verbose=True):
    model.train()
    if sampler != None:
        sampler.set_epoch(epoch)

    metric_train_loss = AverageMeter()
    ametric_data_time = AverageMeter()

    top1 = AverageMeter()
    top5 = AverageMeter()


    end = time.time()
    lr = optimizer.param_groups[0]['lr']
    with tqdm(total=len(dataloader), desc=args.tqdm_desc_head + 'Train Epoch  #{}'.format(epoch), disable=not verbose) as t:
        for batch_idx, (img, target) in enumerate(dataloader):
            if args.cuda:
                img = img.cuda()
                target = target.cuda()

            optimizer.zero_grad()
            output = model(img)
            loss = criterion(output, target)

            loss.backward()
            optimizer.step()

            prec1, prec5 = accuracy(output, target, (1, 5))
            count_try = img.cpu().shape[0]
            top1.update(prec1[0].detach().cpu().item(), count_try)
            top5.update(prec5[0].detach().cpu().item(), count_try)

            metric_train_loss.update(loss.detach().cpu().item(), count_try)
            t.set_postfix({'loss': metric_train_loss.avg, 'lr':lr,  'top1':top1.avg, 'top5':top5.avg })
            t.update(1)

    ametric_data_time.update(time.time() - end)

    if log_writer:
        log_writer.add_scalar('train/loss', metric_train_loss.avg, epoch)

    try:
        print_string = 'Epoch: [{0}][{1}/{2}]\t'.format(epoch, batch_idx, len(dataloader))
        print_string += 'Data time {data_time.val:.3f} ({data_time.avg:.3f} Now:{Now})\t'.format(data_time=ametric_data_time,
Now=datetime.datetime.now())
        print_string += 'Loss {loss.val:.4f} ({loss.avg:.4f})'.format(loss=metric_train_loss)
        print_string += 'Accuracy top1:{top1.avg:.4f}, top5:{top5.avg:.4f}'.format(top1=top1, top5=top5)
        print(print_string)
    except:
        pass

    return metric_train_loss.avg
```

학습시 수행

```python
def valid(args, dataloader, sampler,  model, criterion, epoch, log_writer=None, verbose=True):
    metric_train_loss = AverageMeter()
    ametric_data_time = AverageMeter()
    top1 = AverageMeter()
    top5 = AverageMeter()
    if sampler != None:
        sampler.set_epoch(epoch)
    model.eval()
    end = time.time()
    args.list_preds = []
    args.list_target = []
    args.count_correct = 0
    with torch.no_grad():
        with tqdm(total=len(dataloader), desc=args.tqdm_desc_head + '{} Epoch  #{}'.format( args.run_phase, epoch), disable=not verbose)
as t:
            for i, (img, target, path_img, aug_name ) in enumerate(dataloader):
                if args.cuda:
                    img = img.cuda()
                    target = target.cuda()
                output = model(img)
                loss = criterion(output, target)
                prec1, prec5 = accuracy(output, target, (1, 5))

                if ( prec1[0].detach().cpu().item() != 100.):
                    if args.run_phase == 'valid': print(f'<------ class valid fail file: {path_img[0]}, aug_name:{aug_name}')
                preds = output.data.max(dim=1, keepdim=True)[1]
                count_correct = preds.eq(target.data.view_as(preds)).cpu().sum()
                list_preds = preds.view(-1).tolist()
                args.list_preds = list_preds
                args.list_target = target.detach().cpu().tolist()
                args.count_correct = count_correct.item()
                args.path_img = path_img
                count_try = img.cpu().shape[0]
                top1.update(prec1[0].detach().cpu().item(), count_try)
                top5.update(prec5[0].detach().cpu().item(), count_try)

                metric_train_loss.update(loss.detach().cpu().item(), count_try)
                t.set_postfix({'loss': metric_train_loss.avg, 'top1': top1.avg, 'top5': top5.avg})
                t.update(1)
    ametric_data_time.update(time.time() - end)
    if log_writer:
        log_writer.add_scalar('validation/loss', metric_train_loss.avg, epoch)
    try:
        print_string = 'Epoch: [{0}][{1}/{2}]\t'.format(epoch, i, len(dataloader))
        print_string += 'Data time {data_time.val:.3f} ({data_time.avg:.3f} Now:{Now})\t'.format(data_time=ametric_data_time,
Now=datetime.datetime.now())
        print_string += 'Loss {loss.val:.4f} ({loss.avg:.4f})'.format(loss=metric_train_loss)
        print_string += 'Accuracy top1:{top1.avg:.4f}, top5:{top5.avg:.4f}'.format(top1=top1, top5=top5)
        print(print_string)
    except:
        pass

    return metric_train_loss.avg
```

검증시 수행

3. 실행파일 설명

pill_classifier.py

```
def run_model(args, model, dataloader_train, dataloader_valid, sampler_train, sampler_valid,
criterion,optimizer, epoch_begin, log_writer, verbose=True  ):
    if args.run_phase == 'valid' or args.run_phase == 'test':
        print(time.asctime())
        valid(args, dataloader_valid, sampler_valid,  model, criterion, 0, log_writer, verbose)
        print(time.asctime())
        return

    lr_scheduler = ReduceLROnPlateau(optimizer, mode='min', factor=0.8, patience=2, verbose=True,
threshold=0.0001, threshold_mode='rel', cooldown=3, min_lr=0, eps=1e-08)


    best_perf = 1000
    for epoch in range(epoch_begin, args.epochs):   # ( 0:100)
        # adjust_learning_rate(args, optimizer, epoch)
        # train for one epoch
        perf_indicator = train(args, dataloader_train, sampler_train, model, criterion, optimizer, epoch, log_writer,
verbose)
        if epoch > 10:
            perf_indicator = valid(args, dataloader_valid,sampler_valid, model, criterion, epoch,
log_writer,verbose)
            if (args.gpu == 0):
                print(f'perf_indicator:{perf_indicator} ,  best_perf:{best_perf}')
            if perf_indicator < best_perf:
                model_save(args.model_path, epoch, model, optimizer, args.rank)
                best_perf = perf_indicator
        else:
            model_save(args.model_path, epoch, model, optimizer, args.rank)
            best_perf = perf_indicator

        lr_scheduler.step(perf_indicator)


model = None
criterion = None
optimizer = None
epoch_begin = 0
log_writer = None
```

```
def pill_classifier(args):
    global model, criterion, optimizer, epoch_begin, log_writer
    if args.dataset_valid != None:
        dataloader_valid = DataLoader(args.dataset_valid, batch_size=args.batch_size, shuffle= False,
num_workers=args.num_workers)
    else:
        dataloader_valid = None

    if args.run_phase == 'train' and args.dataset_train != None:
        dataloader_train = DataLoader(args.dataset_train, batch_size=args.batch_size, shuffle=True,
num_workers=args.num_workers)
    else:
        dataloader_train = None

    if model == None :
        log_writer = SummaryWriter(args.dir_log)
        if args.cuda == False or torch.cuda.device_count() == 0  :
            args.gpu = None
        else:
            args.gpu = 0

        args.rank = args.gpu

        cudnn.benchmark = True
        torch.backends.cudnn.deterministic = False
        torch.backends.cudnn.enabled = True

        model = get_pill_model(args)

        # define loss function (criterion) and optimizer
        criterion = torch.nn.CrossEntropyLoss()
        if args.cuda:
            criterion = criterion.cuda()
        optimizer = get_optimizer(args,model)
        epoch_begin, dict_checkpoint, success = model_load(args, model, optimizer)

    run_model(args, model, dataloader_train, dataloader_valid, None, None,  criterion,optimizer, epoch_begin,
log_writer )
```

3. 실행파일 설명

main_cls01_dir.py

```
from pill_classifier import *
from get_cli_args import get_cli_args
from pathlib import Path
from PIL import Image
import os

class Dataset_Dir(Dataset):
    def __init__(self, args, dir_dataset, transform=None, target_transform=None, run_phase='train'):
        self.args = args
        self.dir_dataset = dir_dataset
        self.transform = transform
        self.target_transform = target_transform

        self.list_images = [ png.name  for png in Path(dir_dataset).iterdir() if png.suffix == '.png']
        self.run_phase = run_phase

    def __len__(self):
        return len(self.list_images)

    def __getitem__(self, idx):
        image = Image.open(os.path.join(self.dir_dataset, self.list_images[idx]))
        label = 0
        path_img = self.list_images[idx]
        aug_name = ""

        if self.transform is not None:
            image = self.transform(image)

        if self.target_transform is not None:
            label = self.target_transform(label)
        if self.run_phase == 'valid' or self.run_phase == 'test':
            return image, label, path_img, aug_name
        else:
            return image, label
```

```
if __name__ == '__main__':
    # job = 'hrnet_w64'
    job = 'resnet152'
    args = get_cli_args(job=job, run_phase='test', aug_level=0, dataclass='01')

    print(f'model_path_in is {args.model_path_in}')

    dir_testimage = r'.\dir_testimage'

    args.dataset_valid = Dataset_Dir(args, dir_testimage, transform=transform_normalize,
                                     run_phase='test' if args.run_phase == 'test' else 'valid')
    args.batch_size = len(args.dataset_valid)
    args.verbose = False
    print(f'valid dataset was loaded')

    pill_classifier(args)

    print(args.path_img)
    print(args.list_preds)
    print('job done')
```