# OralCancer Classification 2th - 코드 분석

> Writer : intern1

코드 분석 + PL 사용법 알아보기

## train.py - main

[L71]
시드 고정

```
pl.trainer.seed_everything(seed=255)
```

[L72]
인자들 받기 (`python3 train.py --model efficientnet_b3...`)

```
parser = argparse.ArgumentParser(description='Arguments')
parser.add_argument('--mode', type=str, default='train')
... 열심히 argument 추가 ...
```

[L106]
이건 왜 저렇게 하는지는 의문..?

```
parser = pl.Trainer.add_argparse_args(parser)
parser.add_argument('--save_dir', type=str, required=True)
..,
hparams = parser.parse_args()
```

## train.py - train()

[L15]
큰 차이 없지만 다루기 쉬운 dict으로.

```
if not isinstance(params, dict):
    args = vars(params)
```

[L24]
Stacking Ensemble을 위한 `KFold`
sklearn에 구현되어 있다.

```
kfold = KFold(n_splits=args['fold_num'], shuffle=True)
for i, (train_index, val_index) in enumerate(kfold.split(path_names, y=None)):
```

[L28]
데이터셋과 데이터로더

```
train_dataset = OcancerDataset_train(**data_kwargs)
train_collate = OcancerCollate_train(**collate_kwargs)
train_loader = DataLoader(train_dataset, batch_size = args['batch_size'],
shuffle=True, collate_fn=train_collate, num_workers=args['num_workers'])

val...
```

하나씩 알아보자.

---

## Dataloader.py - OcancerDataset_train

[L78]
`Dataset`을 상속받아 구현한다.

- Rotation -> Flip -> Affine -> Bright 순서로 50% 확률로 변환을 수행한다. (`Augmentation - Others`에 해당)

이는 이미지 한 장에 대한 `Augmentation` 이다.

```
argument_list = []
if self.rotation and np.random.uniform(0, 1) > 0.5:
    argument_list.append(self.randomRotation)
...
shuffle(argument_list)
argument_list += [self.resize, torchvision.transforms.ToTensor()]
for arg_func in argument_list:
    img = arg_func(img)
img = img
```

## Dataloader.py - OcancerCollate_train

[L133]
`train()`에서 `train_loader`의 `collate_fn`에 들어갈 함수를 `__call__`을 이용해 구현한 클래스이다. `collate_fn`은 배치 처리를 위해 쓰이며 여기서는 다음과 같은 두 가지 목적을 가지고 있다.

- cutmix, mixup 등 **여러** 이미지를 이용한 Augmentation
- 정답 레이블 -> 모델의 아웃풋 형태로 변환

[L155]
`multiclass` 여부에 따라 정답 배열을 잘 만든다.

```
labels = torch.zeros(size=(len(batch), self.label_num))
labels[torch.arange(0, len(labels)), batch_labels] = 1
batch_labels = labels
```

## Dataloader.py - Mixup

[L14]

```
Mixup(x, y, alpha=1.0, device='cpu')
```

배치 단위의 x와 y를 입력 받아 각 이미지에 다른 랜덤 이미지의 x와 y를 섞는다.

$\lambda * x + (1 - \lambda) * x[index]$

$\lambda * y + (1 - \lambda) * y[index]$

## Dataloader.py - Cutmix

[L47]

```
Cutmix(x, y, alpha=1.0, devide='cpu')
```

Mixup을 통해 Cutout을 개선시킨 요즘 Augmentation

## Dataloader.py - OcancerDataset_val, OcancerCollate_val

[L176] [L198]
validation data loader을 위한 Augmentation 없는 구현

## 다시 train.py - train()

`train_loader`와 `val_Loader`선언

```
train_loader = DataLoader(train_dataset, batch_size = args['batch_size'],
shuffle=True, collate_fn=train_collate, num_workers=args['num_workers'])
val_loader = DataLoader(val_dataset, batch_size=args['batch_size'],
collate_fn=val_collate, num_workers=args['num_workers'])
        callbacks = []
```

[L37]
cli 환경에서 로그 찍기 위한 콜백 함수 (PyTorch Ligntning에서 지원한다)
학습용 로스(bce)와 실제 평가 함수(roc) 둘다 찍고 있다.

```
if args['mode'] == 'train':
    checkpoint_callback_loss = ModelCheckpoint(
        monitor='val_loss',
    ...
    callbacks = [checkpoint_callback_loss, checkpoint_callback_roc]
```

[L57]
특성 모아 Trainer 만들자.

```
trainer = pl.Trainer(**trainer_kwargs)
```

가장 중요한 모델

```
model = BinaryClassificationModel(params.__dict__, i)
```

## Modelloader.py - BinaryClassificationModel

[L88]
`pl.LightningMoudle` 을 상속받아 구현되었다.
Pretrained_model에 BinaryClassifier나 MultiClassifier을 통과시킨다.

## Modelloader.py - BinaryClassifier

[L10]
`nn.Module`을 상속받아 구현되었다.
주워온 Pretrained Model 끝단에 붙이는 Classifier
`model_name` (pretrain model) 이 무엇이냐에 따라 input 크기 맞춰서 init한다.
output 크기는 2

## Modelloader.py - MultiClassifier

[L38]
`nn.Module`을 상속받아 구현되었다.
output 크기가 3
좀 더 길고 깊게 구현되었다.

## 다시 trian.py - train()

[L61]
auto lr 관련 ..? `new_lr.suggestion`을 바로 모델에 대입하는 이유는 잘 모르겠다.
auto lr은 좋은 lr을 찾아나가는 과정 아닌가? 문서보면 코드 같게 생겨서 더 모르겠다.

```
if args['auto_lr_find']:
    new_lr = trainer.tuner.lr_find(model, train_loader)
    params.lr = new_lr.suggestion()
    model = BinaryClassificationModel(params.__dict__)
    print('found learning rate : {:04f}'.format(new_lr.suggestion()))
```

```
trainer.fit(model, train_loader, val_loader)
```

inference 관련 부분은 생략