지능화 캡스톤 프로젝트

프로젝트 #1 결과 발표

2023. 4. 26

충북대학교 산업인공지능학과

[3조] 김현기, 원윤재

INDUSTRIAL AI RESEARCH CENTER

수행방법 및 업무분장

수행방법

- 같은 회사에 재직중이어서 수시로 얘기하여 업무 분장 및 수행
-

업무분장 및 기여도

이름	수행내용	비고
김현기	• 데이터 증량 • 주제발표	
원윤재	• 코딩/학습 • 결과발표	

데이터셋

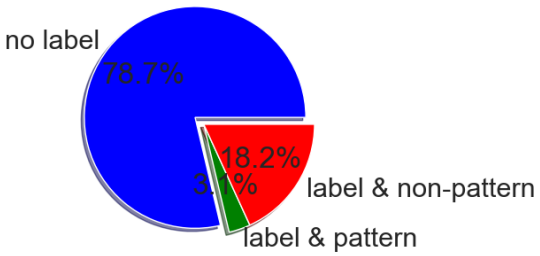
원본 데이터셋 → 특정 해상도 추출 (25,27),(26,26) → 결함 케이스별 분류 (Failure type)

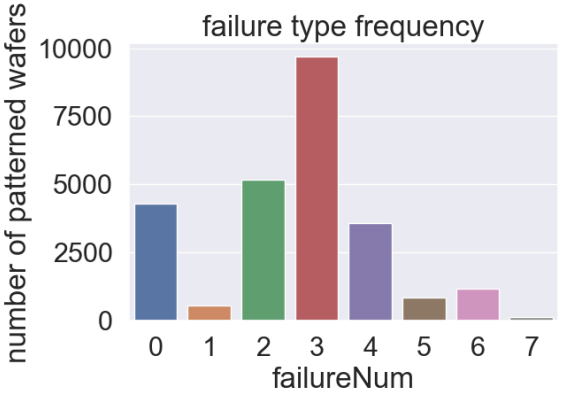
결함 케이스별 분류 (Failure type) → Encoder

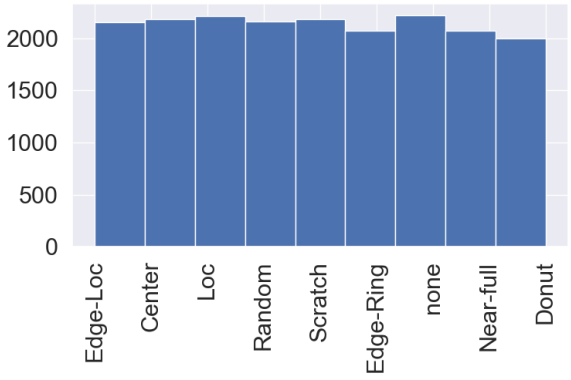
Encoder
Optimizer : Adam
Loss : MSE
Batch Size : 16
Learning Rate : 0.001
Epoch : 10

→ 결함 케이스별 증량 *결함별 약 2천개 →

Decoder
Optimizer : Adam
Loss : MSE
Batch Size : 16
Learning Rate : 0.001
Epoch : 10

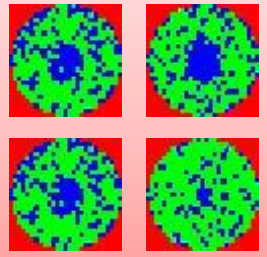




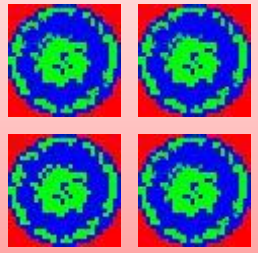


데이터셋

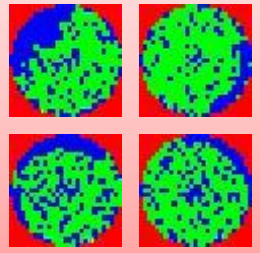
Center



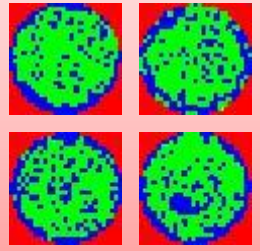
Dounut



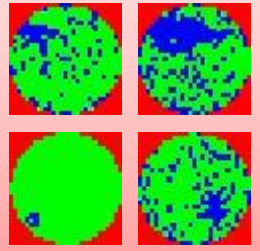
Edge-Loc



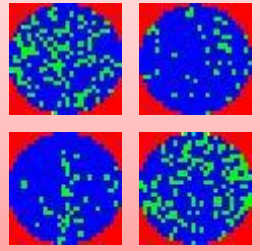
Edge-Ring



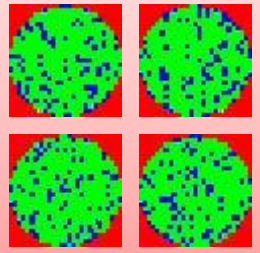
Loc



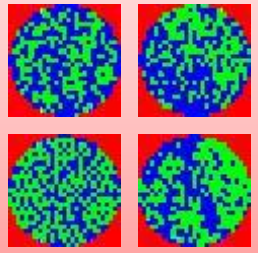
Near-full



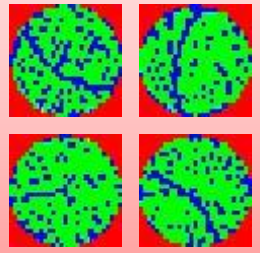
None



Random

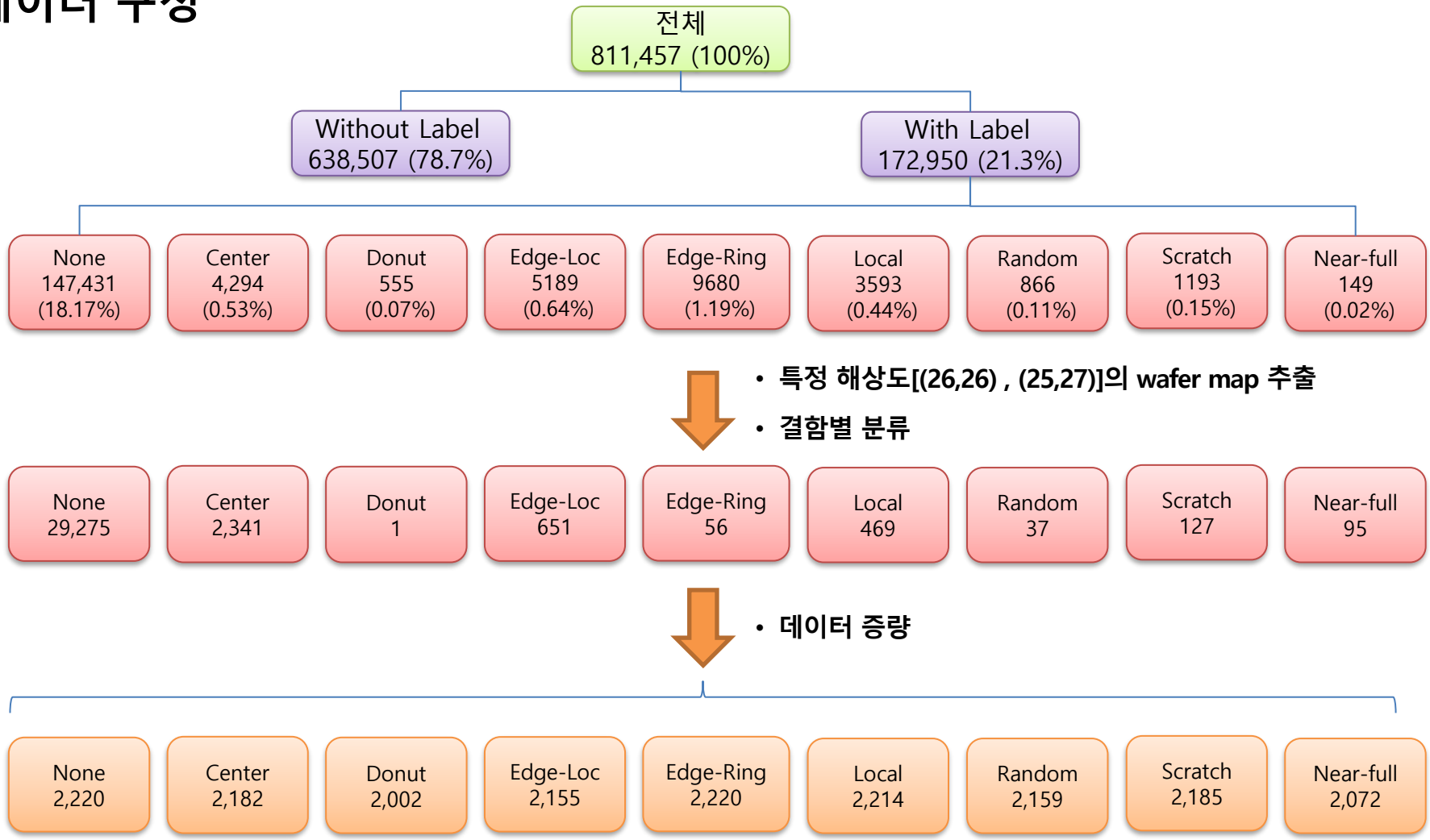


Scratch



데이터셋

데이터 구성



Train : Validation : Test = 64 : 16 : 20

연구 방법

데이터 증량에 따른 모델 성능 비교

- Data 증량 수에 따른 결과값 비교

Case 1 : 결함 클래스별 1,000개 증량

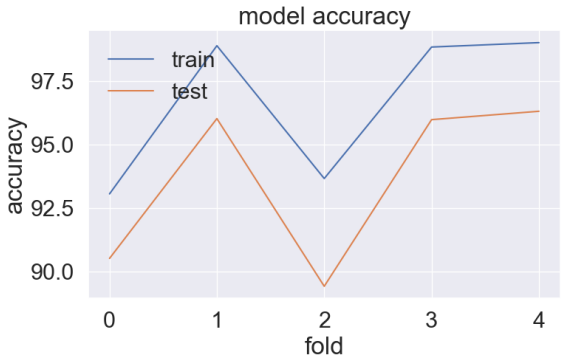
Average Training Loss: 0.083
Average Test Loss: 0.091
Average Training Acc: 96.70
Average Test Acc: 93.66
학습 소요시간 : 약 7분 소요

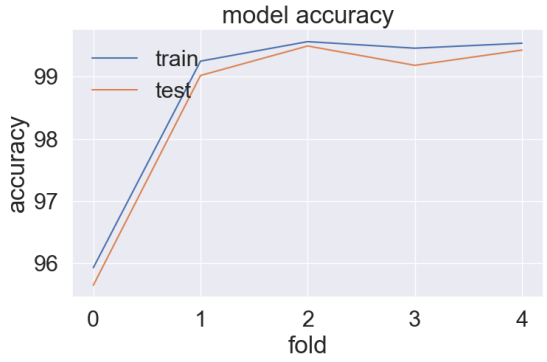
Case 2 : 결함 클래스별 2,000개 증량

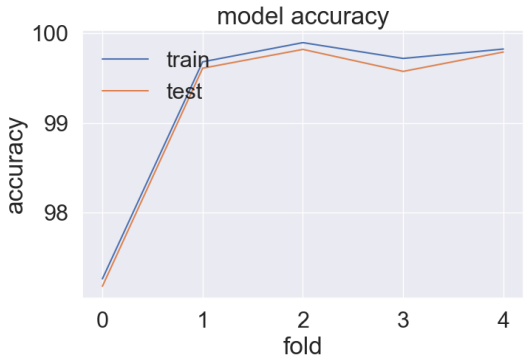
Average Training Loss: 0.031
Average Test Loss: 0.039
Average Training Acc: 98.75
Average Test Acc: 98.56
학습 소요시간: 약 15분 소요

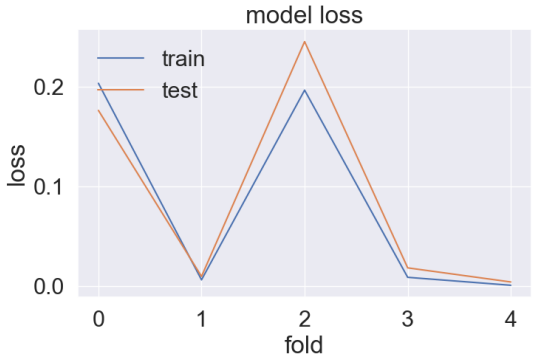
Case 3 : 결함 클래스별 3,000개 증량

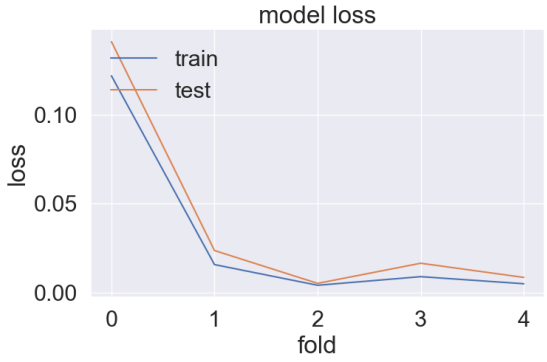
Average Training Loss: 0.025
Average Test Loss: 0.028
Average Training Acc: 99.28
Average Test Acc: 99.20
학습 소요시간: 약 28분 소요

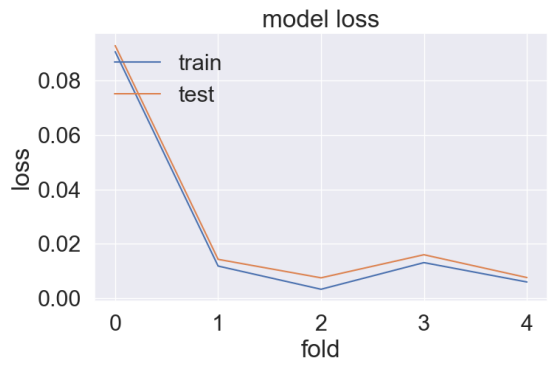






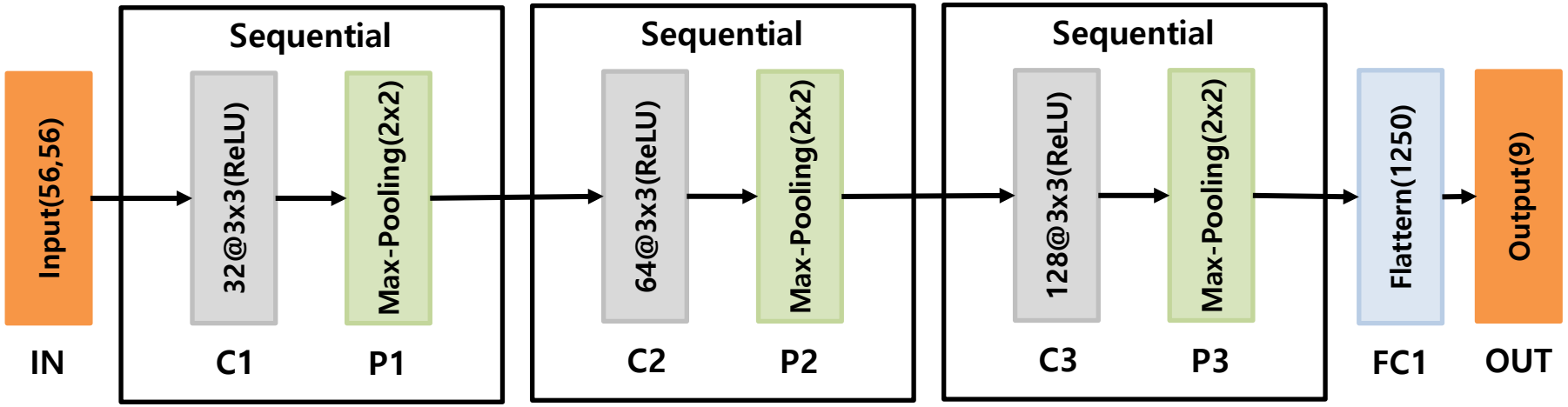






CNN 구조 / 전이학습

CNN 구조



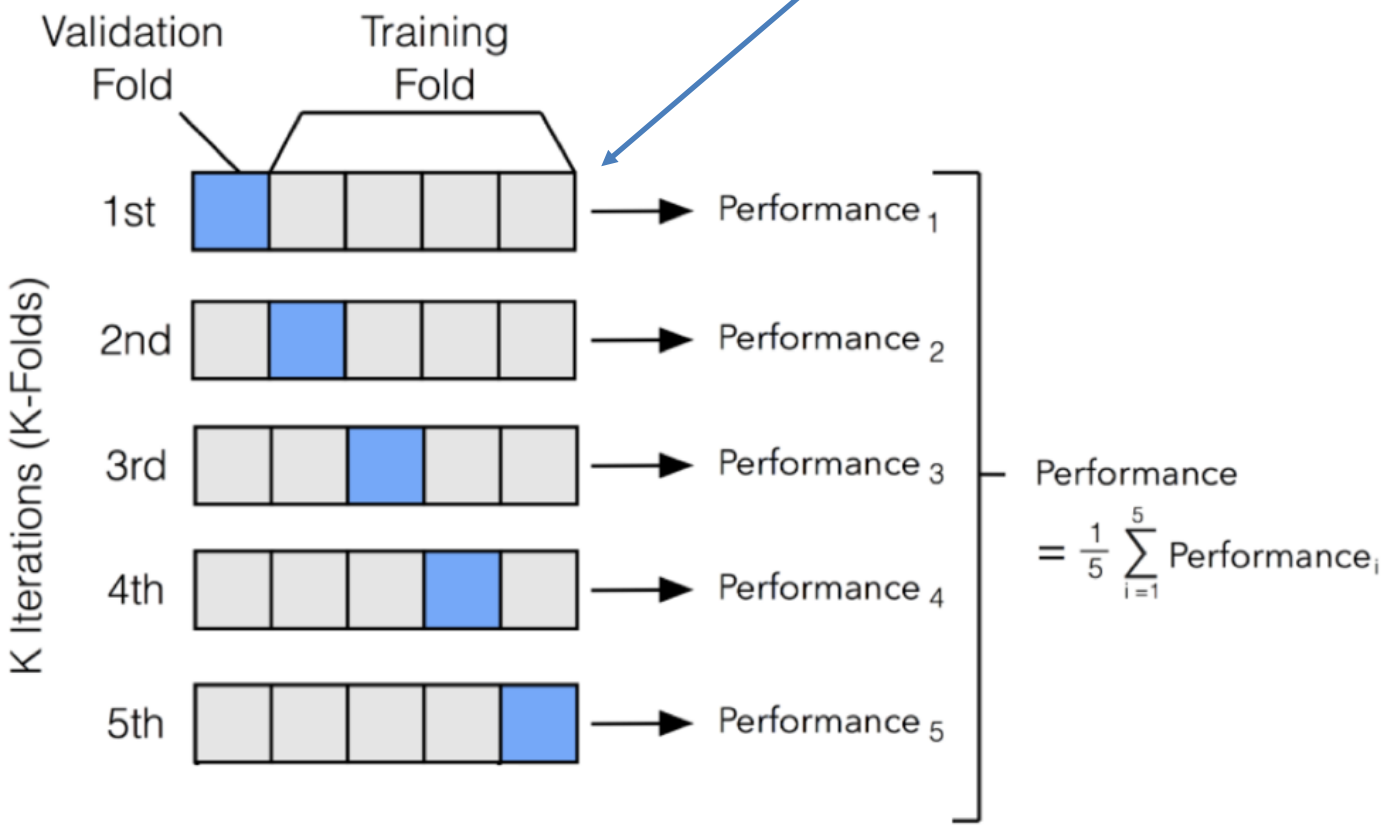

```
CNN(
  (layer1): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer3): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=1, dilation=1, ceil_mode=False)
  )
  (fc1): Linear(in_features=8192, out_features=1250, bias=True)
  (layer4): Sequential(
    (0): Linear(in_features=8192, out_features=1250, bias=True)
    (1): ReLU()
  )
  (fc2): Linear(in_features=1250, out_features=9, bias=True)
)
```

CNN 구조 / 전이학습

주요 코드 및 실행 결과

- K-fold Cross Validation [교차검증]

Train : Validation : Test = 64 : 16 : 20



CNN 구조 / 전이학습

주요 코드 및 실행 결과

- K-fold Cross Validation [교차검증]

```
splits = KFold(n_splits=5, shuffle = True, random_state = 42)

for fold, (train_idx,val_idx) in enumerate(splits.split(np.arange(len(dataset)))):
    print('Fold {}'.format(fold + 1))

    train_sampler = SubsetRandomSampler(train_idx)
    test_sampler = SubsetRandomSampler(val_idx)
    train_loader = DataLoader(dataset, batch_size=args['BATCH_SIZE'], sampler=train_sampler, drop_last=True)
    test_loader = DataLoader(dataset, batch_size=args['BATCH_SIZE'], sampler=test_sampler, drop_last=True)
    history = {'train_loss': [], 'test_loss': [],'train_acc':[],'test_acc':[]}

    for epoch in range(args['NUM_EPOCH']):
        train_loss, train_correct=train_epoch(CNN,train_loader,criterion,optimizer)
        test_loss, test_correct=valid_epoch(CNN,test_loader,criterion)
        train_loss = train_loss / len(train_loader.sampler)
        train_acc = train_correct / len(train_loader.sampler) * 100
        test_loss = test_loss / len(test_loader.sampler)
        test_acc = test_correct / len(test_loader.sampler) * 100

        print("Epoch:{}/{} AVG Training Loss:{:.3f} AVG Test Loss:{:.3f} AVG Training Acc {:.2f} % AVG Test Acc {:.2f} %".format(epoch + 1,
                                                                    args['NUM_EPOCH'],
                                                                    train_loss,
                                                                    test_loss,
                                                                    train_acc,
                                                                    test_acc))
        history['train_loss'].append(train_loss)
        history['test_loss'].append(test_loss)
        history['train_acc'].append(train_acc)
        history['test_acc'].append(test_acc)

    foldperf['fold{}'.format(fold+1)] = history
```

CNN 구조 / 전이학습

주요 코드 및 실행 결과

- K-fold Cross Validation [교차검증]

```
Fold 1
Epoch:1/20 AVG Training Loss:1.272 AVG Test Loss:0.402 AVG Training Acc 64.04 % AVG Test Acc 83.62 %
Epoch:2/20 AVG Training Loss:0.311 AVG Test Loss:0.208 AVG Training Acc 88.03 % AVG Test Acc 92.06 %
Epoch:3/20 AVG Training Loss:0.175 AVG Test Loss:0.175 AVG Training Acc 93.56 % AVG Test Acc 93.54 %
...
Epoch:19/20 AVG Training Loss:0.004 AVG Test Loss:0.035 AVG Training Acc 99.60 % AVG Test Acc 98.88 %
Epoch:20/20 AVG Training Loss:0.006 AVG Test Loss:0.044 AVG Training Acc 99.60 % AVG Test Acc 98.68 %

Fold 2
Epoch:1/20 AVG Training Loss:0.019 AVG Test Loss:0.031 AVG Training Acc 99.10 % AVG Test Acc 98.49 %
Epoch:2/20 AVG Training Loss:0.179 AVG Test Loss:0.034 AVG Training Acc 94.97 % AVG Test Acc 98.75 %
Epoch:3/20 AVG Training Loss:0.026 AVG Test Loss:0.022 AVG Training Acc 98.84 % AVG Test Acc 98.96 %
...
Fold 5
Epoch:1/20 AVG Training Loss:0.003 AVG Test Loss:0.005 AVG Training Acc 99.60 % AVG Test Acc 99.56 %
Epoch:2/20 AVG Training Loss:0.004 AVG Test Loss:0.003 AVG Training Acc 99.54 % AVG Test Acc 99.64 %
Epoch:3/20 AVG Training Loss:0.111 AVG Test Loss:0.042 AVG Training Acc 97.61 % AVG Test Acc 98.21 %
...
Epoch:12/20 AVG Training Loss:0.001 AVG Test Loss:0.004 AVG Training Acc 99.64 % AVG Test Acc 99.58 %
Epoch:13/20 AVG Training Loss:0.001 AVG Test Loss:0.005 AVG Training Acc 99.64 % AVG Test Acc 99.58 %
Epoch:14/20 AVG Training Loss:0.002 AVG Test Loss:0.005 AVG Training Acc 99.63 % AVG Test Acc 99.58 %
Epoch:15/20 AVG Training Loss:0.002 AVG Test Loss:0.005 AVG Training Acc 99.64 % AVG Test Acc 99.53 %
Epoch:16/20 AVG Training Loss:0.001 AVG Test Loss:0.006 AVG Training Acc 99.64 % AVG Test Acc 99.53 %
Epoch:17/20 AVG Training Loss:0.001 AVG Test Loss:0.005 AVG Training Acc 99.64 % AVG Test Acc 99.53 %
Epoch:18/20 AVG Training Loss:0.001 AVG Test Loss:0.006 AVG Training Acc 99.64 % AVG Test Acc 99.53 %
Epoch:19/20 AVG Training Loss:0.001 AVG Test Loss:0.006 AVG Training Acc 99.64 % AVG Test Acc 99.53 %
Epoch:20/20 AVG Training Loss:0.001 AVG Test Loss:0.006 AVG Training Acc 99.64 % AVG Test Acc 99.53 %
```

학습 방법

PC 사양

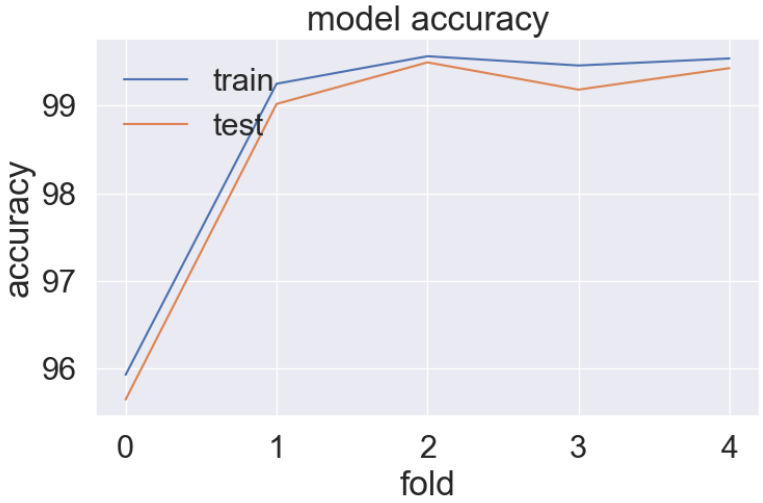
CPU : Intel(R) Core(TM) i9-9960X CPU @ 3.10GHz
RAM : 32GB
GPU : NVIDIA GeForce RTX 2080 TI 10GB
학습 시간: 약 15분

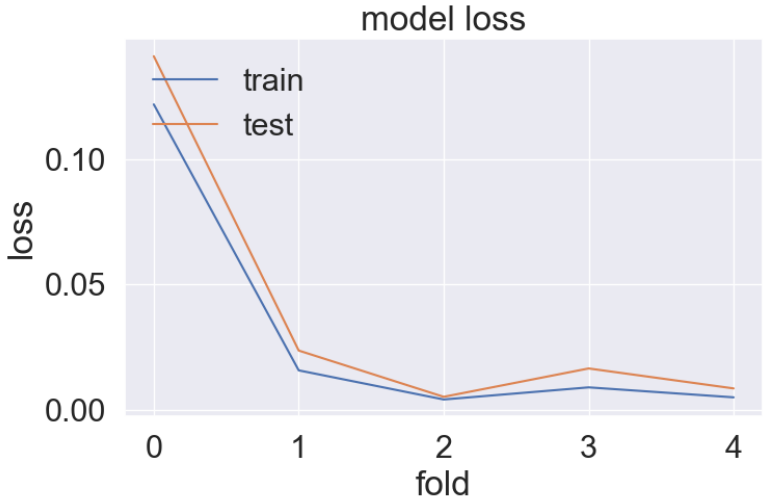
결과

Average Training Loss: 0.031
Average Test Loss: 0.039
Average Training Acc: 98.75
Average Test Acc: 98.56

하이퍼 파라미터

Optimizer : Adam
Loss : torch.nn.CrossEntropyLoss
Batch Size : 256
Learning Rate : 0.005
Epoch : 20
K-fold:{
    n-split: 5
    shuffle: True
    random_state: 42
}





결과 및 토의

분류 성능

- *Confusion matrix 및 평가지표*
- *구체적인 분석*

```
def eval_model(model, dataloader):
    classes = ['Center', 'Donut', 'Edge-Loc', 'Edge-Ring', 'Loc', 'Near-full', 'Random',
'Scratch', 'none']

    model.eval()
    confusion_matrix = torch.zeros(len(classes), len(classes))
    with torch.no_grad():
        for inputs, targets in dataloader:
            inputs, targets = inputs.to(DEVICE), targets.to(DEVICE)
            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for t, p in zip(targets.view(-1), preds.view(-1)):
                confusion_matrix[t.long(), p.long()] += 1

    class_accuracy = {}
    for i in range(len(classes)):
        class_accuracy[classes[i]] = 100 * confusion_matrix[i, i] /
confusion_matrix[i, :].sum()

    return class_accuracy
```

```
[실행결과]
{
 'Center': tensor(15.9896),
 'Donut': tensor(51.2500),
 'Edge-Loc': tensor(nan),
 'Edge-Ring': tensor(nan),
 'Loc': tensor(nan),
 'Near-full': tensor(nan),
 'Random': tensor(nan),
 'Scratch': tensor(nan),
 'none': tensor(nan)
}
```

결과 및 토의

토의 및 개선점

- PyTorch 학습 환경 구성에 많은 시간이 소요 됨

- CNN에 대한 이해 부족

- 다양한 모델을 적용하여 분석하지 못함

- 클래스별 모델평가를 확인하지 못함

감사합니다