이미지로 작물의 생육기간 예측하기 (pytorch 기반 딥러닝 프로젝트)

1조

고선욱
유상준
장현영

PRESENTATION CONTENTS

1 서론 및 주제 선정 배경

1-1 연구 배경 (스마트팜)

스마트팜에 접목된 딥러닝 기술

식물 병해 예측, 식물 분류, 수확량 예측, 물체 탐지 등

다양한 분야에 활용되고 있는 스마트팜 딥러닝 기술

실내 온실, 식당에도 접목된 스마트팜

스마트팜 기반 최적의 생육환경을 갖춘 실내 온실에서 수확되는 농작물

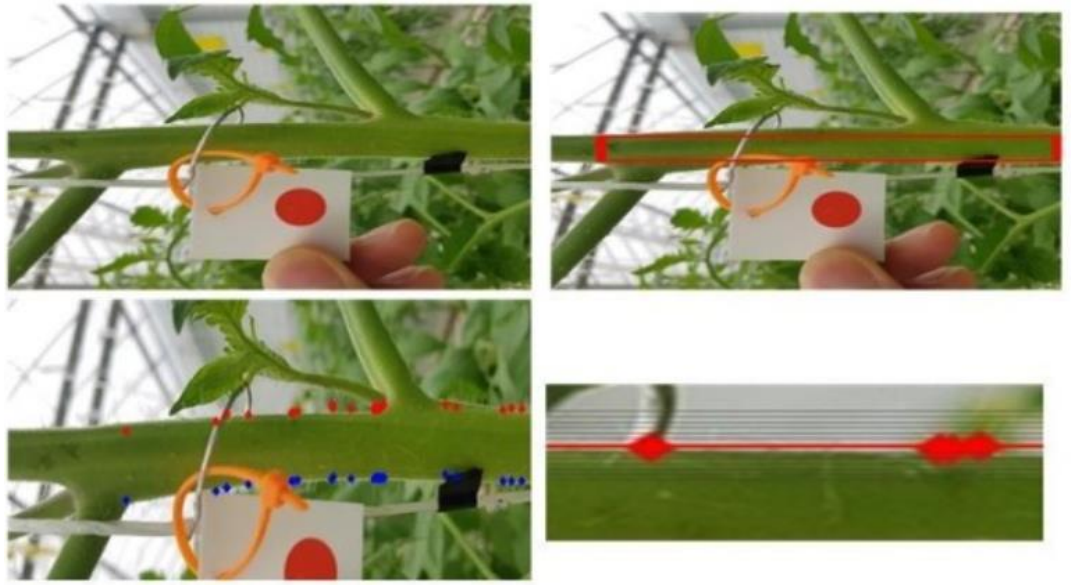
AI, 빅데이터를 통해 일반 농지의 약 40배 생산성

1-2 주제 선정 및 목표

이미지만으로 생육기간을 예측할 수 있을까?

이미지에서 생육 기간에 따른 작물의 크기, 줄기의 외곽선을 탐지하여 생육지표로 활용 가능

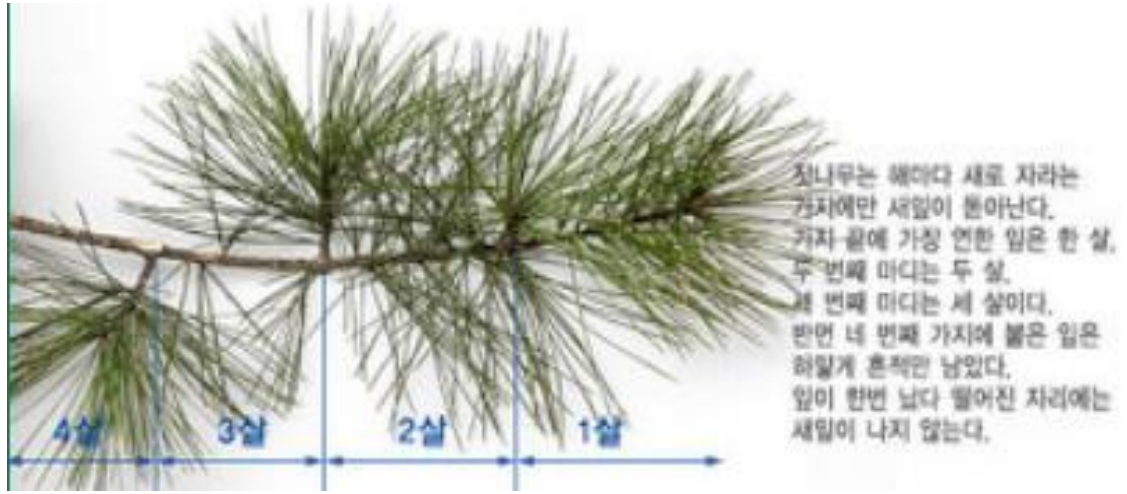
딥러닝 이미지 학습을 통해 생육기간을 예측



목표 : 생육기간 예측 모델 개발 및 분석

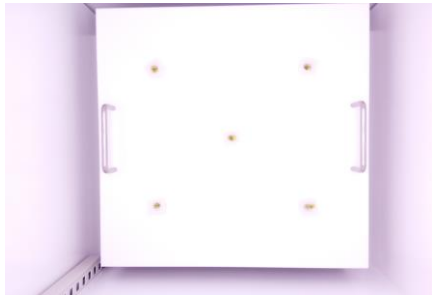
- 두 쌍의 이미지를 입력 값으로 제공 받아 다양한 딥러닝 예측 모델을 구현

- 농업 종사자의 직관적 경험으로 판단하는 생육기간을 딥러닝 모델을 통해 정량적 수치를 계산하기

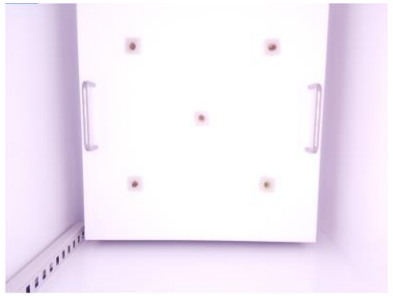


2 데이터 소개 및 문제 정의

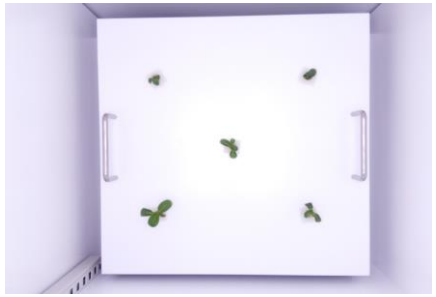
2-1 데이터 소개



< BC_01 – DAT01 >



< LT_10 – DAT01 >



< BC_01 – DAT15 >

< LT_10 – DAT17 >

< BC_01 – DAT40 >

< LT_10 – DAT39 >

train data set

BC – 청경채, 9개의 폴더, 각 폴더당 30~40개의 이미지
(파일명 DAT*.png는 일수(30~40일)로 추정됨)

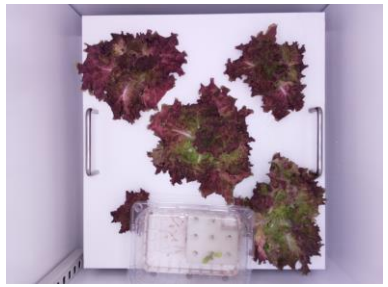
LT – 적상추, 10개의 폴더, 각 폴더당 30~40개의 이미지
파일명 DAT*.png는 일수(30~40일)로 추정됨)

2-1 데이터 소개

< idx_BC_1112_00090 >

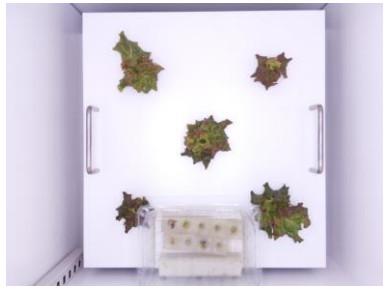
< idx_LT_1003_00592 >

< idx_BC_1112_00239 >



< idx_LT_1003_00102 >

< idx_BC_1112_00914 >



< idx_LT_1003_00972 >

test data set

BC – 청경채, 3개의 폴더로 구성 (1088, 1100, 1112)
각각 59, 46, 34개의 이미지 파일 포함,
파일명은 무작위 숫자로 추정

LT – 적상추, 3개의 폴더로 구성 (1003, 1088, 1089)
각각 64, 49, 55 개의 이미지 파일 포함,
파일명은 무작위 숫자로 추정

2-1 데이터 소개

	idx	time_delta
0	0	-1
1	1	-1
2	2	-1
3	3	-1
4	4	-1
...	...	...
3955	3955	-1
3956	3956	-1
3957	3957	-1
3958	3958	-1
3959	3959	-1

3960 rows × 2 columns

< sample_submission.csv >

	idx	before_file_path	after_file_path
0	0	idx_LT_1003_00341	idx_LT_1003_00154
1	1	idx_LT_1003_00592	idx_LT_1003_00687
2	2	idx_BC_1100_00445	idx_BC_1100_00840
3	3	idx_BC_1112_00229	idx_BC_1112_00105
4	4	idx_LT_1088_00681	idx_LT_1088_00698
...	...	...	...
3955	3955	idx_BC_1100_00110	idx_BC_1100_00525
3956	3956	idx_LT_1089_00006	idx_LT_1089_00442
3957	3957	idx_BC_1100_00511	idx_BC_1100_00132
3958	3958	idx_BC_1088_00353	idx_BC_1088_00196
3959	3959	idx_BC_1100_00527	idx_BC_1100_00640

3960 rows × 3 columns

< test_data.csv >

sample_submission.csv

idx와 time_delta(기간차이)로 구성, 모델로 예측값 갱신

test_data.csv

idx, before_file, after_file로 구성

데이터는 test dataset의 파일명으로 구성되어 있어 Path로 설정해야 함

모두 3960개의 idx로 구성되어 있음

2개의 이미지 파일을 Before, after 한 쌍으로 묶어서 3960개 케이스를 생성

※ 이미지 크기가 크다보니 용량이 커서 224*224 resize 후 사용

2-2 문제 정의 및 가설

- Test_data의 정확한 일 수 차이를 모르기 때문에 RMSE로 loss 판단
- 모델의 크기와 gpu 성능, 시간적 한계를 고려하여 기존 사용모델을 이용하기도 함
- Timm 라이브러리(이미지 딥러닝 학습 모듈)에서 제공한 훈련된 일부 모델 사용
- Test data set에서 1088,1112 등 동일한 폴더 내에 있는 이미지끼리 pair로 묶임
- Train set에서도 bc_01 등 같은 폴더 내에 있는 이미지는 동일품종 동일식물로 판단
- 따라서 train data set에서 pair 조합을 진행할 때 같은 폴더 내에서만 진행
- Colab 환경에서 원본 이미지 size 변환에 매우 오랜 시간이 걸려
  원본 데이터를 224*224로 resize 선작업 진행

3 모델 설명 및 구현

3-0

데이터 전처리

데이터 불러오기 (1)

```python
def extract_day(images):
    day = int(images.split('.')[-2][-2:])
    return day

def make_day_array(images):
    day_array = np.array([extract_day(x) for x in images])
    return day_array
```

- 파일명 DATxx에서 일수 array화
- 동일 폴더에서 랜덤으로 2개씩 추출하여 pair
- 추출한 2개의 데이터의 파일 경로와 차이 값 (delta) 삽입

```python
def make_combination(length, species, data_frame, direct_name):
    before_file_path = []
    after_file_path = []
    time_delta = []

    for i in range(length):

        direct = random.randrange(0,len(direct_name))
        temp = data_frame[data_frame['version'] == direct_name[direct]]

        sample = temp[temp['species'] == species].sample(2)
        after = sample[sample['day'] == max(sample['day'])].reset_index(drop=True)
        before = sample[sample['day'] == min(sample['day'])].reset_index(drop=True)

        before_file_path.append(before.iloc[0]['file_name'])
        after_file_path.append(after.iloc[0]['file_name'])
        delta = int(after.iloc[0]['day'] - before.iloc[0]['day'])
        time_delta.append(delta)

    combination_df = pd.DataFrame({
        'before_file_path': before_file_path,
        'after_file_path': after_file_path,
        'time_delta': time_delta,
    })

    combination_df['species'] = species

    return combination_df
```

3-0

데이터 전처리

데이터 불러오기 (2)

```
# BC 폴더와 LT 폴더에 있는 하위 폴더를 저장한다.
bc_direct = glob(root_path + '/BC/*')
bc_direct_name = [x[-5:] for x in bc_direct]
lt_direct = glob(root_path + '/LT/*')
lt_direct_name = [x[-5:] for x in lt_direct]

# 하위 폴더에 있는 이미지들을 하위 폴더 이름과 매칭시켜서 저장한다.
bc_images = {key : glob(name + '/*.png') for key,name in zip(bc_direct_name, bc_direct)}
lt_images = {key : glob(name + '/*.png') for key,name in zip(lt_direct_name, lt_direct)}

# 하위 폴더에 있는 이미지들에서 날짜 정보만 따로 저장한다.
bc_dayes = {key : make_day_array(bc_images[key]) for key in bc_direct_name}
lt_dayes = {key : make_day_array(lt_images[key]) for key in lt_direct_name}

bc_dfs = []
for i in bc_direct_name:
    bc_df = pd.DataFrame({
        'file_name':bc_images[i],
        'day':bc_dayes[i],
        'species':'bc',
        'version':i
    })
    bc_dfs.append(bc_df)
```

```
lt_dfs = []
for i in lt_direct_name:
    lt_df = pd.DataFrame({
        'file_name':lt_images[i],
        'day':lt_dayes[i],
        'species':'lt',
        'version':i
    })
    lt_dfs.append(lt_df)

bc_dataframe = pd.concat(bc_dfs).reset_index(drop=True)
lt_dataframe = pd.concat(lt_dfs).reset_index(drop=True)
total_dataframe = pd.concat([bc_dataframe, lt_dataframe]).reset_index(drop=True)

bc_combination = make_combination(5000, 'bc', total_dataframe, bc_direct_name)
lt_combination = make_combination(5000, 'lt', total_dataframe, lt_direct_name)

bc_train = bc_combination.iloc[:4500]
bc_valid = bc_combination.iloc[4500:]

lt_train = lt_combination.iloc[:4500]
lt_valid = lt_combination.iloc[4500:]

train_set = pd.concat([bc_train, lt_train])
valid_set = pd.concat([bc_valid, lt_valid])
```

```
train_dataset = TrainDataset(train_set)
valid_dataset = TestDataset(valid_set)

train_data_loader = DataLoader(train_dataset,
                               batch_size=batch_size,
                               shuffle=True)

valid_data_loader = DataLoader(valid_dataset,
                               batch_size=valid_batch_size)
```

- 경로명에서 파일명, 일수, 작물, 폴더버전을 데이터프레임 화
- 총 경우의 수는 약 14000개지만, 5000개를 sampling하여 조합 추출
- Validation 진행을 위해 9:1 비율로 train과 validation set을 구분

3-1

mobilenet_v2

모델 선정 이유

1. 수업시간에 사용해본 적 있는 모델

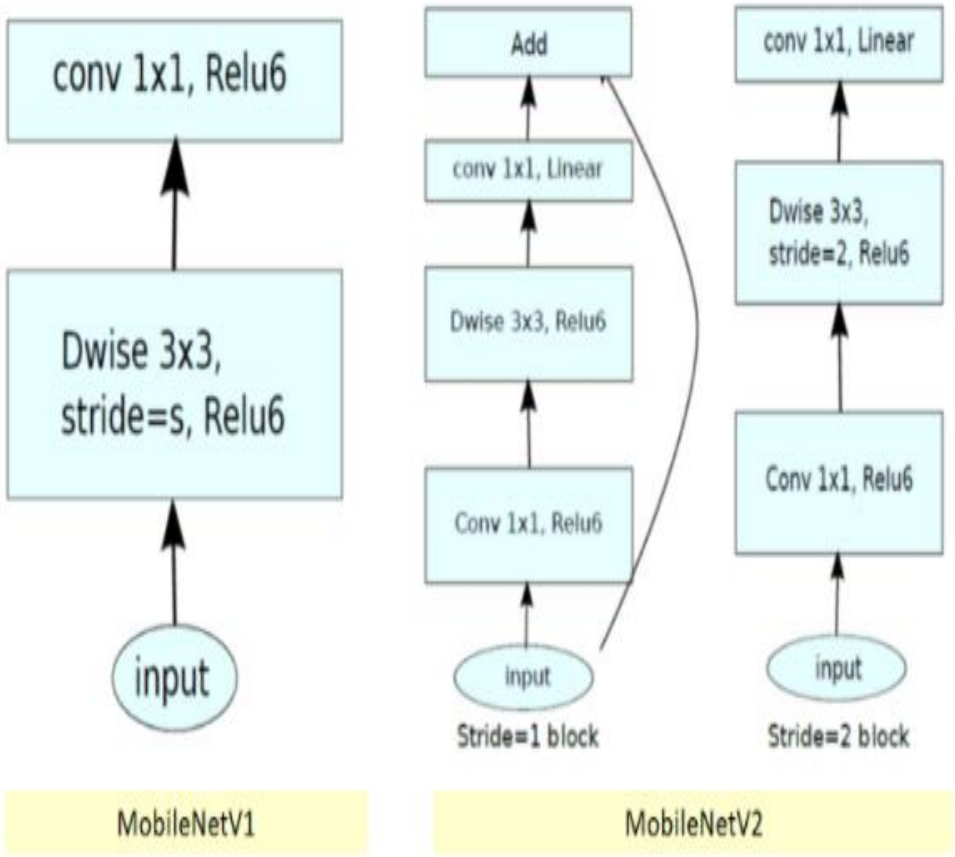
2. params, 연산속도를 획기적으로 줄이면서 적은 파라미터 양으로 우수한 성능

3. 컨볼루션 모듈에서 대형 중간 텐서를 완전 구현 x
- 메모리 공간을 크게 줄일 수 있다는 장점

고성능이 아닌 환경에서 유리

Mobilenet은 컴퓨터 성능이 다소 제한되거나 배터리 퍼포먼스 중요한 상황에서 사용되기 위해 설계된 cnn 구조

convolution -> depthwise separable convolution
(3*3 filter 사용 시 약 8~9배 연산량의 감소)



3-1

mobilenet_v2

Class 선언

```
import torch
from torch import nn
from torchvision.models import mobilenet_v2


class CompareCNN(nn.Module):

    def __init__(self):
        super(CompareCNN, self).__init__()
        self.mobile_net = mobilenet_v2(pretrained=True)
        self.fc_layer = nn.Linear(1000, 1)

    def forward(self, input):
        x = self.mobile_net(input)
        output = self.fc_layer(x)
        return output
```

```
class CompareNet(nn.Module):

    def __init__(self):
        super(CompareNet, self).__init__()
        self.before_net = CompareCNN()
        self.after_net = CompareCNN()

    def forward(self, before_input, after_input):
        before = self.before_net(before_input)
        after = self.after_net(after_input)
        delta = before - after
        return delta
```

3-1

mobilenet_v2

초기 파라미터 설정

```
lr = 1e-5
epochs = 10
batch_size = 64
valid_batch_size = 50

model = CompareNet().to(device)
```

증강, 전이학습, 미세조정 없이 Mobilenet_v2로 학습 시

다양한 Batch size, epochs를 시도했으나 큰 유의미한 차이를 보이지 못함

Test rmse : 7.25

3-2

regnet

모델 선정 이유

Facebook ai research에서 개발한 imagenet

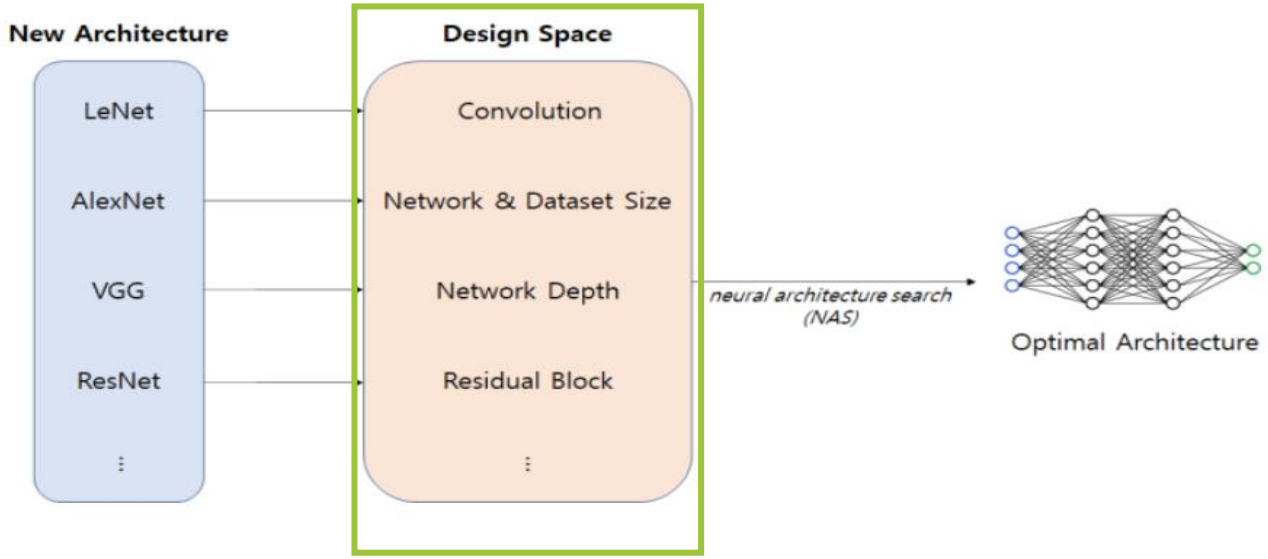
불필요한 element를 하나씩 제거하면서 만들어진 모델 regnet

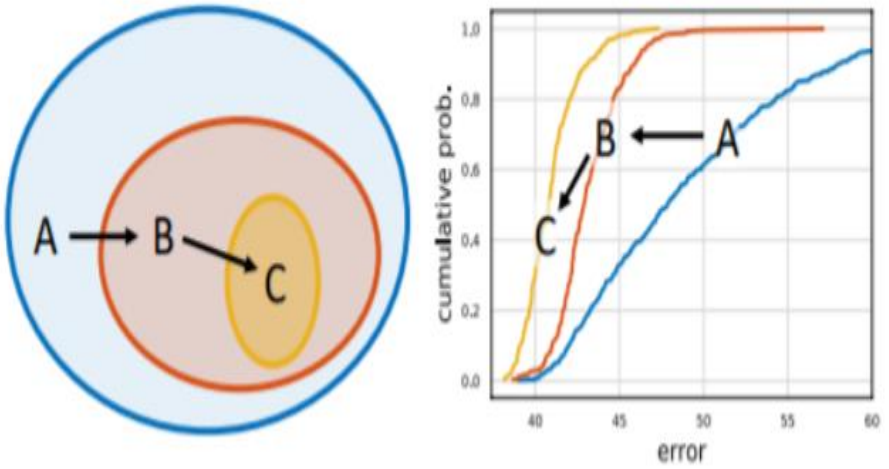
비교적 최신기술로 efficientnet과 비교 시 우수한 성능 나타냄

크기가 커진 DESIGN SPACE(설계 요소)들 중

Error에 대한 설명을 가장 잘 나타내는 요소인

부분을 찾아 나머지 공간을 소거하는 방식





3-2 regnet

regnetx_004

Timm library를 통해 regnet 중 컴퓨터 성능을 고려

available encoders가 상대적으로 적은 regnetx_004로 진행

```
import timm
from pprint import pprint
model_names = timm.list_models(pretrained=True)
```

Encoder	Weights	Params, M
timm-regnetx_002	imagenet	2M
timm-regnetx_004	imagenet	4M
timm-regnetx_006	imagenet	5M
timm-regnetx_008	imagenet	6M
timm-regnetx_016	imagenet	8M
timm-regnetx_032	imagenet	14M

```
class CompareCNN(nn.Module):

    def __init__(self):
        super(CompareCNN, self).__init__()
        self.regnet = model = timm.create_model('regnetx_004', pretrained=True, num_classes=1)

    def forward(self, input):
        x = self.regnet(input)
        return x
```

Test rmse : 7.40

3-3

resnet50

모델 선정 이유

- 1. 수업시간에 사용 해 본적 있는 모델

- 2. 거대한 Data set인 이미지넷에서 미리 훈련되었다는 장점을 이용
  - 전이학습 + 미세조정 용이
  - 시간 단축 가능

모델 처리 방법

- 1. 분류 문제를 회귀 문제로 Change
  - 출력층의 layer : 1000 -> 1

- 2. before model, after model 총 두가지가 필요
  - 식물의 생육 기간을 예측하고, 두 값의 차이를 return

3-3

resnet50

Class 선언

```python
import torch
from torch import nn
from torchvision.models import resnet50


class CompareCNN(nn.Module):

    def __init__(self):
        super(CompareCNN, self).__init__()
        self.resnet = resnet50(pretrained=True)
        self.fc_layer = nn.Linear(1000, 1)

    def forward(self, input):
        x = self.resnet(input)
        output = self.fc_layer(x)
        return output
```

```python
class CompareNet(nn.Module):

    def __init__(self):
        super(CompareNet, self).__init__()
        self.before_net = CompareCNN()
        self.after_net = CompareCNN()

    def forward(self, before_input, after_input):
        before = self.before_net(before_input)
        after = self.after_net(after_input)
        delta = before - after
        return delta
```

3-3

resnet50

학습 과정

<파라미터 설정 및 모델 선언>

```
device = 'cuda:0' if torch.cuda.is_available() else 'cpu'
lr = 1e-5
epochs = 20
batch_size = 32
valid_batch_size = 50

model = CompareNet().to(device)
```

<optimizer 선언>

```
optimizer = optim.Adam(model.parameters(), lr=lr)
```

<Train, Val datasets 준비>

```
bc_combination = make_combination(5000, 'bc', total_dataframe, bc_direct_name)
lt_combination = make_combination(5000, 'lt', total_dataframe, lt_direct_name)

bc_train = bc_combination.iloc[:4500]
bc_valid = bc_combination.iloc[4500:]

lt_train = lt_combination.iloc[:4500]
lt_valid = lt_combination.iloc[4500:]

train_set = pd.concat([bc_train, lt_train])
valid_set = pd.concat([bc_valid, lt_valid])



train_dataset = TrainDataset(train_set)
valid_dataset = TestDataset(valid_set)

train_data_loader = DataLoader(train_dataset,
                               batch_size=batch_size,
                               shuffle=True)

valid_data_loader = DataLoader(valid_dataset,
                               batch_size=valid_batch_size)
```

3-3

resnet50

학습 과정

<train 과정>

```
for epoch in tqdm(range(epochs)):
    for step, (before_image, after_image, time_delta) in tqdm(enumerate(train_data_loader)):
        before_image = before_image.to(device)
        after_image = after_image.to(device)
        time_delta = time_delta.to(device)

        optimizer.zero_grad()
        logit = model(before_image, after_image)
        train_loss = (torch.sum(torch.abs(logit.squeeze(1).float() - time_delta.float())) /
                      torch.LongTensor([batch_size]).squeeze(0).to(device))
        train_loss.backward()
        optimizer.step()

        if step % 15 == 0:
            print('\n====================loss=====================')
            print(f'\n====================EPOCH: {epoch}=====================')
            print(f'\n====================step: {step}=====================')
            print('MAE_loss : ', train_loss.detach().cpu().numpy())
```

<validation set eval 과정>

```
valid_losses = []
with torch.no_grad():
    for valid_before, valid_after, time_delta in tqdm(valid_data_loader):
        valid_before = valid_before.to(device)
        valid_after = valid_after.to(device)
        valid_time_delta = time_delta.to(device)


        logit = model(valid_before, valid_after)
        valid_loss = (torch.sum(torch.abs(logit.squeeze(1).float() - valid_time_delta.float())) /
                      torch.LongTensor([valid_batch_size]).squeeze(0).to(device))
        valid_losses.append(valid_loss.detach().cpu())


print(f'VALIDATION_LOSS MAE : {sum(valid_losses)/len(valid_losses)}')
checkpoint = {
    'model': model.state_dict(),

}

torch.save(checkpoint, 'resnet50_v3.pt')
```

3-3

resnet50

학습 과정

<train 과정 예시>

```
100%|████████████████████| 20/20 [00:12<00:00, 1.58it/s]
VALIDATION_LOSS MAE : 1.9605789184570312
282/? [03:36<00:00, 1.64it/s]

====================loss====================

====================EPOCH: 6====================

====================step: 0====================
MAE_loss :  1.0429683

====================loss====================

====================EPOCH: 6====================

====================step: 15====================
MAE_loss :  1.3808079

====================loss====================

====================EPOCH: 6====================

====================step: 30====================
MAE_loss :  1.1285472
```

3-3

resnet50

학습 과정

<test set load, evaluation 과정>

```
test_set = pd.read_csv('./drive/MyDrive/open_224/test_dataset/test_data.csv')
test_set['l_root'] = test_set['before_file_path'].map(lambda x: './drive/MyDrive/open_224/test_dataset/' + x.split('_')[1] + '/' + x.split('_')[2])
test_set['r_root'] = test_set['after_file_path'].map(lambda x: './drive/MyDrive/open_224/test_dataset/' + x.split('_')[1] + '/' + x.split('_')[2])
test_set['before_file_path'] = test_set['l_root'] + '/' + test_set['before_file_path'] + '.png'
test_set['after_file_path'] = test_set['r_root'] + '/' + test_set['after_file_path'] + '.png'

test_dataset = TestDataset(test_set, is_test=True)
test_data_loader = DataLoader(test_dataset,
                              batch_size=64)
```

```
test_value = []
with torch.no_grad():
    for test_before, test_after in tqdm(test_data_loader):
        test_before = test_before.to(device)
        test_after = test_after.to(device)
        logit = model(test_before, test_after)
        value = logit.squeeze(1).detach().cpu().float()

        test_value.extend(value)
```

3-3

resnet50

Test set loss 확인

- Test set 에 대한 label 이 주어지지 않았다.
- DAYCON 사이트에 제출해서 확인

3-3

resnet50

추가 시도 1 : five crop 사용

< 가설 >

- 회전, 뒤집기, 아핀 변환 등을 통해 여러 raw한 상태의 이미지를 증강했을 때, test set에 대한 loss 가 효율적으로 감소 함을 확인
- 이미지의 특성 상 좌상, 우상, 중앙, 우하, 좌하로 배치
- randomcrop을 통해 이미지가 부분화 되었을 때 보다 실제 작물의 정보가 확실하게 반영되었을 때 불필요한 이미지의 noise를 담지 않아 더 좋은 결과를 낼 것이라고 생각

3-3

resnet50

추가 시도 1 : five crop 사용

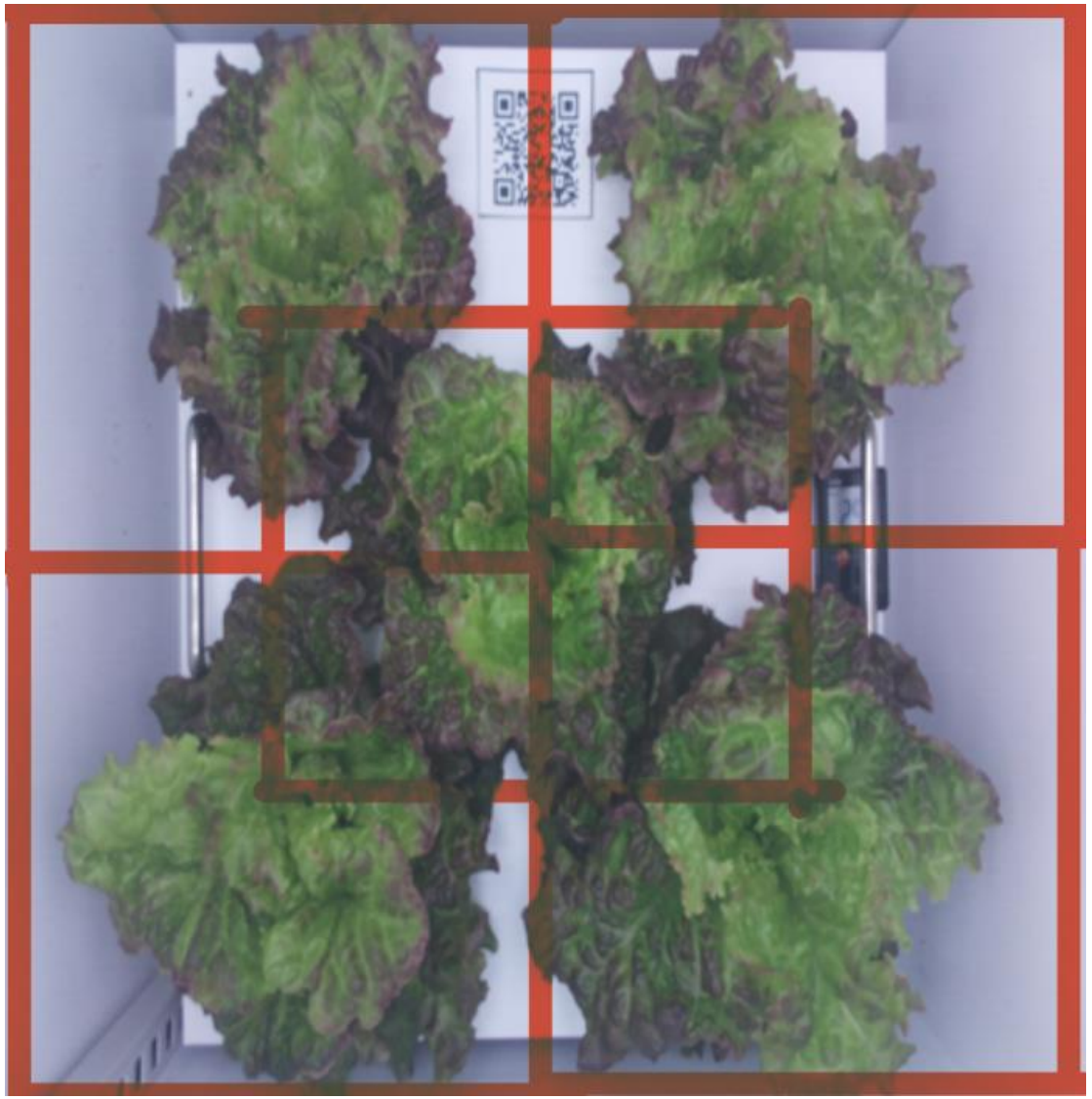
< 설계 >

- 이미지 증강의 방식은 유지하되, crop의 방식만 fivecrop으로 바꾸고, transforms.compose를 유지하여 학습을 시킴
- transforms.FiveCrop(120)
- 업로드한 실제 이미지 사이즈가 224x224이므로 fivecrop을 하더라도 적어도 112x112이상으로 잡아야 최소 한 작물에 대해 담을 수 있으므로 crop size를 120으로 잡고 진행

3-3

resnet50

< 예시 >



3-3

resnet50

추가 시도 1 : five crop 사용

< 코드설명 >

- image를 fivecrop하게 되면 기존 입력 가중치가 4차원에서 crop별로 나누어져, crop 위치에 따른 차원이 하나 늘어나 5차원이 되어 4차원 형태로 유지하거나 변환이 필요
- 랜덤으로 5개의 위치 중 1개의 위치에 대한 crop을 추출하기 위한 함수 작성

```
def randomFromFiveCrops(crops):
    randomIndex = random.randint(0,len(crops)-1)
    return crops[randomIndex]

class TrainDataset(Dataset):
    def __init__(self, combination_df, is_test=None):
        self.combination_df = combination_df
        self.transform = transforms.Compose([
            transforms.FiveCrop(120),
            transforms.Lambda(lambda crops: randomFromFiveCrops(crops)),
            transforms.Resize(224),
            transforms.RandomHorizontalFlip(p=0.5),  # 수평 뒤집기
            transforms.RandomVerticalFlip(p=0.5), # 수직 뒤집기
            transforms.RandomAffine((-20, 20)), # 아핀 변환: 선형 변환에서 이동 변환까지 포함
            transforms.RandomRotation(90), # 90도 회전
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ])
```

3-3

resnet50

추가 시도 1 : five crop 사용

< 결과 >

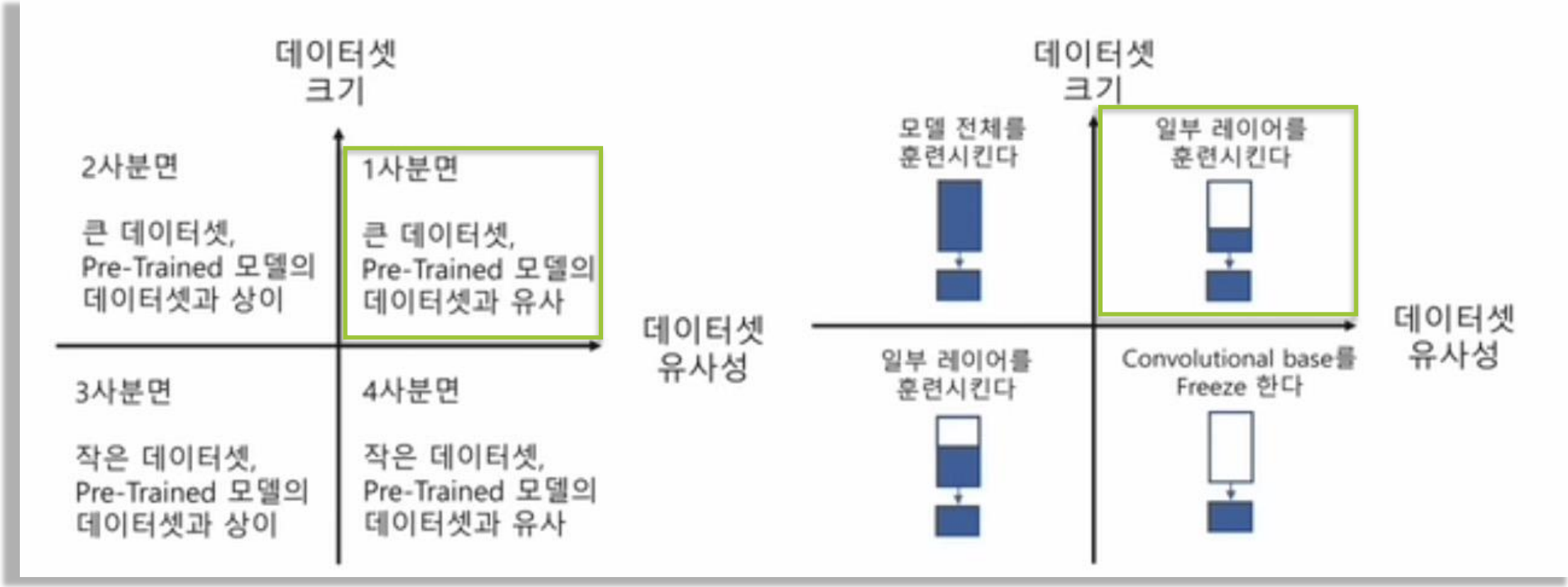
생각보다 오차가 줄어들지 않았다

- 함수의 구성이 잘못되었다. 반환값이 같은 crop위치에 대한 이미지만 반환되므로 데이터셋의 구성이 5crop이 아니라 5crop중 1crop에 대한 데이터셋에 대한 model이 학습됨

- Test RMSE: 6.618

3-3 # resnet50

추가 시도 2 : 미세조정



3-3

resnet50

추가 시도 2 : 미세조정

< 가설 >

- 우리의 데이터는 resnet50에서 학습한 이미지와 성질이 크게 다르지 않고, 이미지 증강으로 인해 많은 학습용 데이터를 확보했다

- 이미 학습된 모델인 resnet50은 크게 보면 10개의 layer로 이루어져 있다

- 현재 우리는 마지막 fc layer인 Linear(1000,1)만을 학습하는 과정을 이용했다.

- 출력층에 가까운순으로 일부 layer의 weights를 우리가 가진 데이터로 학습 시켜조금 더 나은 성능을 기대

3-3

resnet50

추가 시도 2 : 미세조정

< 코드설명 >

```
class CompareNet(nn.Module):

    def __init__(self):
        super(CompareNet, self).__init__()
        self.before_net = CompareCNN()
        self.after_net = CompareCNN()

        before_model = resnet50(pretrained=True)                          # 미세조정 위해서 추가
        before_model.fc_layer = nn.Linear(1000,1)
        before_modules = list(before_model.children())[:-1]               # 미세조정 위해서 추가
        self.before_feature_extract_model = nn.Sequential(*before_modules) # 미세조정 위해서 추가

        after_model = resnet50(pretrained=True)                           # 미세조정 위해서 추가
        after_model.fc_layer = nn.Linear(1000,1)
        after_modules = list(after_model.children())[:-1]                 # 미세조정 위해서 추가
        self.after_feature_extract_model = nn.Sequential(*after_modules) # 미세조정 위해서 추가


    def forward(self, before_input, after_input):
        before = self.before_net(before_input)
        after = self.after_net(after_input)
        delta = before - after
        return delta
```

- CompareNet 클래스 재 작성
- Before image와 After image의 생육 기간을 구하기 위해 따로 모델 설정
- model.children()을 이용해 layer 가져와서 받아주기
  - 추후 미세조정 코드에 사용

3-3

resnet50

추가 시도 2 : 미세조정

< 코드설명 >

```
before_model_ft = model.before_feature_extract_model
ct = 0
for child in before_model_ft.children():
    ct += 1
    if ct < 6:
        for param in child.parameters():
            param.requires_grad = False

after_model_ft = model.after_feature_extract_model
ct = 0
for child in after_model_ft.children():
    ct += 1
    if ct < 6:
        for param in child.parameters():
            param.requires_grad = False
```

** 5번째 layer까진 weights를 Freeze **

- 이후 layer 부터 학습하며 가중치를 갱신하기 위한 코드

- 몇 번째 layer부터 학습하기 시작할 것인지 여러 번 테스트 결과 최적의 결과는 5번째 layer부터
- Test RMSE: 5.35

4 전체 모델 성능 비교

4 전체 모델 성능 비교

기준 : DAYCON 사이트 제출
test data set loss score (rmse)

1. movilenetv2 model : 7.25
2. regnet model : 7.40
3. regnet model + augmentation : 7.59
4. resnet model : 7.04
5. resnet model + augmentation : 5.98
6. resnet model + augmentation : 5.35
   + fine tuning

4 전체 모델 성능 비교

Best Model : 13위 Rank

기준 : DAYCON 사이트 제출
test data set loss score (rmse)

			3.95158	77	8일 전
2			4.41614	43	6일 전
3			4.63286	40	18일 전
4			4.66251	47	10일 전
5		재색	4.73668	55	8일 전
6			4.994	55	12일 전
7			5.08741	42	7일 전
8			5.09783	47	13일 전
9		4k	5.17796	67	14일 전
10		dx	5.25605	46	12일 전
11			5.2874	44	16일 전
12		엣헴	5.33856	28	12일 전
13	욱이	욱이	5.35296	13	30분 전

5 한계 및 보완점

5 한계점, 보완할 점

1. 다양한 이미지 size를 활용하지 못한 점 (only 224*224)
   - 메모리에 올리기 위해 이미지 size를 임의 조정했기에 따라오는 데이터 손실
2. 전반적인 주제에 대한 도메인 지식 부족
   - 만약 이미지를 효율적으로 crop하는 방법에 대한 아이디어가 있었다면…
3. 전이학습, 미세조정 시 레이어를 조금 더 세세하게 만져보지 못한 점
   - 다양한 모델을 이용한 전이학습, 미세조정도 시도해 보지 못한 점
4. Fivecrop 이용시 겪었던 문제점들
   - 가중치 벡터의 차원 일치 문제
5. 딥러닝 모델의 학습시간에 따른 여러가지 시간적 제약

번외

GitHub, Notion page link

- Notion : https://www.notion.so/seonwook97/1-ada66cda850540eabade5a11b0bfa7dc

- GitHub : https://github.com/hyun-young/nklcb_DL_project

감사합니다 :)