

프로젝트형 AI 서비스 개발

생육 환경 최적화 경진대회

2조 자라나라 청경채



contents

①

프로젝트 개요

②

팀 구성 및 역할

③

데이터 소개

④

수행 절차
및 모델 소개

⑤

데이터 결과

⑥

결론

01

프로젝트 개요

Dacon

생육 환경 최적화 경진대회

1. 배경

4차 산업혁명의 시대를 맞아 농업 분야에서도 스마트팜 등 IT 기술을 널리 사용하여 더욱 효율적으로 작물을 재배하고 있습니다.

나아가, AI를 기반으로 작물의 효율적인 생육을 위한 최적의 환경을 도출한다면 식물 재배에 큰 도움이 되지 않을까요?

2. 주제

[Algorithm]

청경채 사진과 환경 데이터를 활용한 잎면적 예측 알고리즘 개발

3. 주체/주관

주최 / 주관: KIST 강릉분원

운영: 데이콘



Image

	위도	경도
18.60000038		
18.60000038	41.3	
18.60000038	41.2999992	
18.5	41.0999984	
18.60000038	41.2000007	
18.5	41.4000015	
18.5	41.5999984	
18.5		
18.5		

Meta

y_name	leaf_wid
CASE01_01.png	45.0
CASE01_02.png	59.0
CASE01_03.png	72.0
CASE01_04.png	85.0
CASE01_05.png	102.0
CASE01_06.png	123.0
CASE01_07.png	150.0
CASE01_08.png	180.0

Label

Dacon

생육 환경 최적화 경진대회

4. 평가산식

: NMAE (Normalized MAE)

Normalized Mean Absolute Error (or
Coefficient of Variation of MAE)

$$NMAE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{MAE(\mathbf{y}, \hat{\mathbf{y}})}{\frac{1}{n} \sum_{i=1}^n |y_i|} = \frac{MAE(\mathbf{y}, \hat{\mathbf{y}})}{mean(|\mathbf{y}|)}$$

NMAE는 scale이 다른 데이터셋의 MAE
비교에 용이

정규화된 MAE이기 때문에 편향되지 않아
과소평가와 과대평가 모두에 대해 유사한
해석을 제공

```
def NMAE(y_true, y_pred):
    error = y_true - y_pred
    absolute_error = tf.abs(error)
    return tf.reduce_mean(absolute_error) / tf.reduce_mean(y_true)

def nmae(y_true, y_pred):
    score = tf.py_function(func=NMAE, inp=[y_true, y_pred], Tout=tf.float32, name='nmae')
    return score
```

02

팀 구성 및 역할

팀 구성 및 역할

자라나라 청경채

INFJ-A



팀장 김인후

시계열 데이터
(RNN, GRU, LSTM),
Multi Input Model

ISFP-A



팀원 이희경

Data pipeline,
Finetuning,
Image Segmentation

INTP-T



팀원 장소희

데이터전처리,
Masking 면적 계산
Pretrain Net

ESFP-T



팀원 유영재

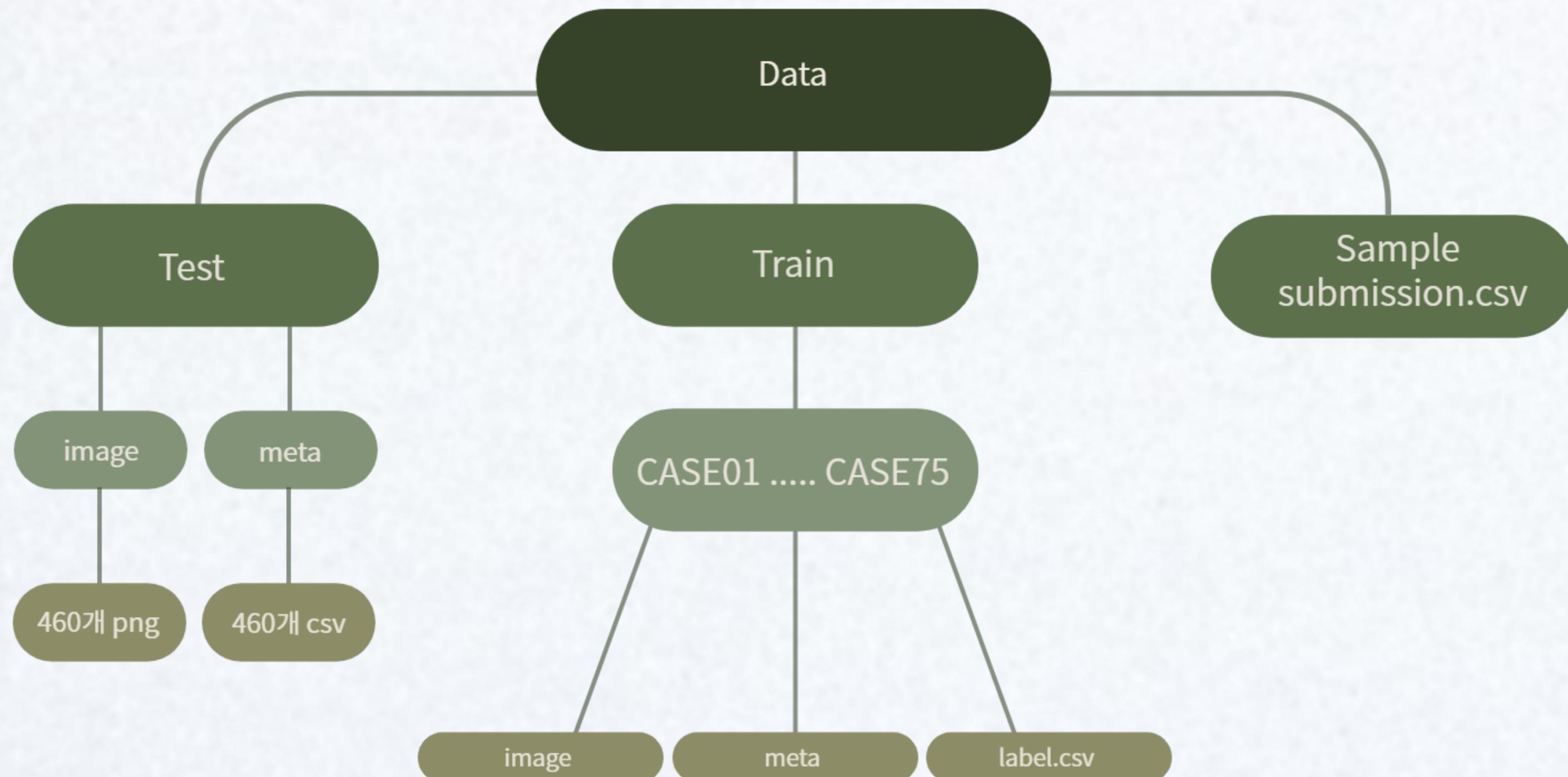
데이터전처리,
Pretrain Net,
Image Segmentation

03

데이터 소개

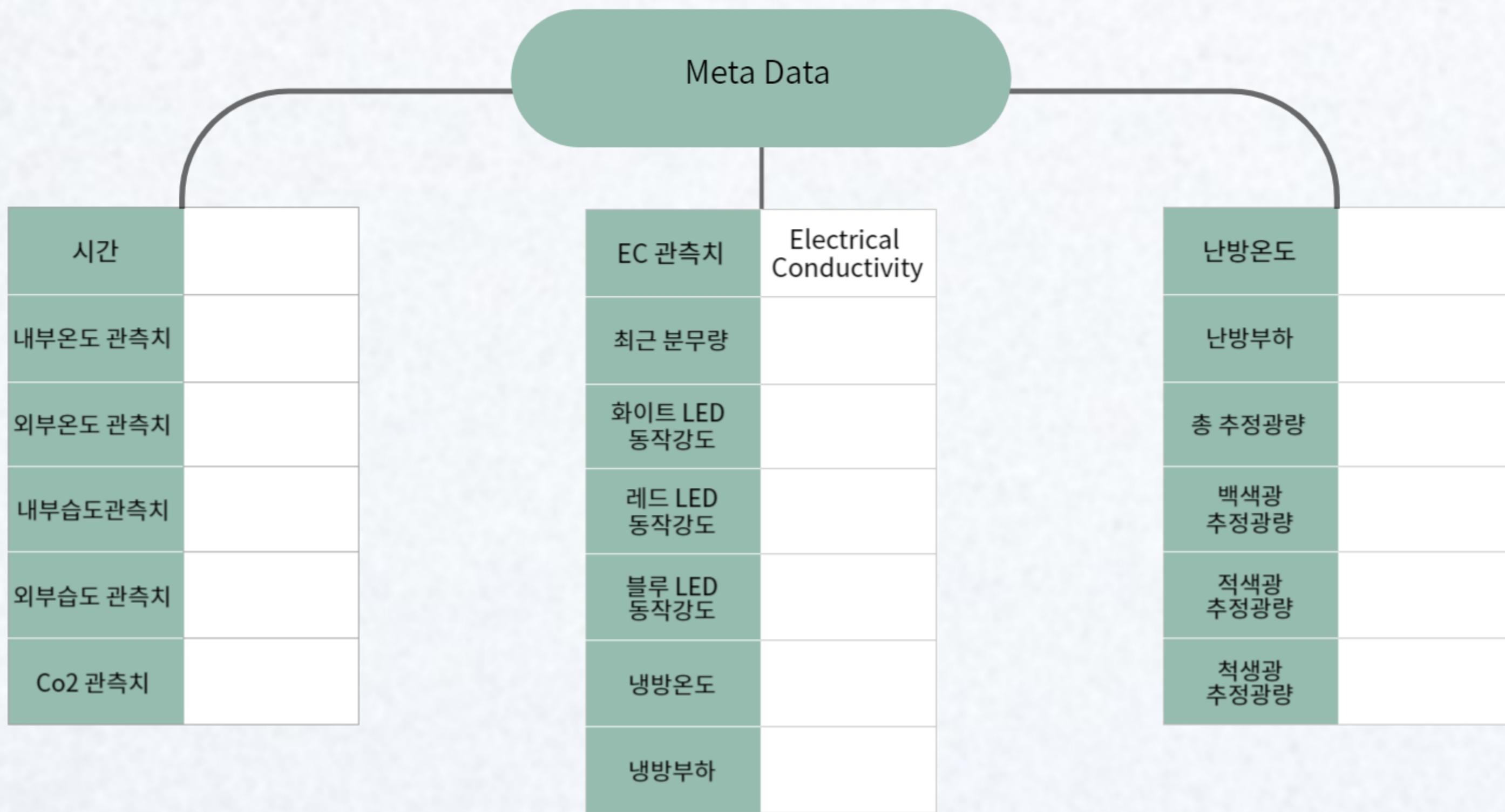
DATA FILE STRUCTURE

데이터파일 구조



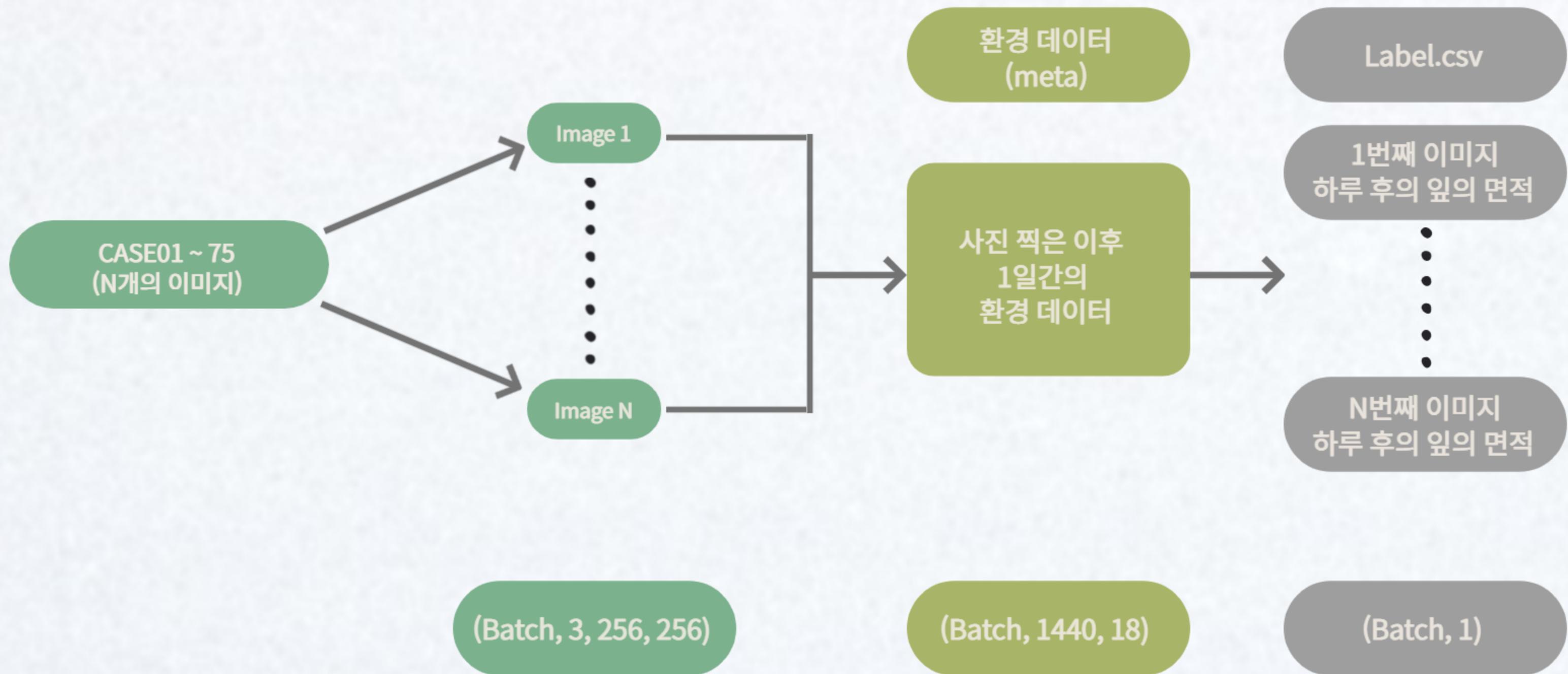
Data File Structure

meta data 구조



DATA FILE STRUCTURE

학습데이터의 흐름

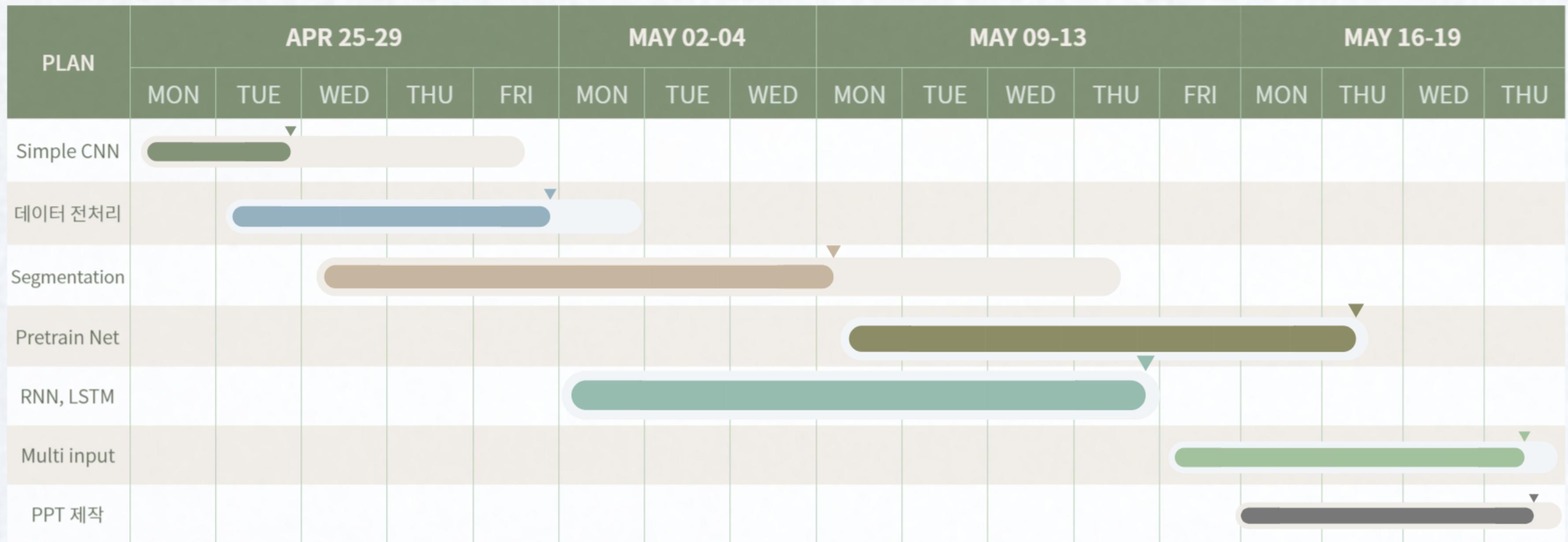


04

수행 절차 및 모델 소개

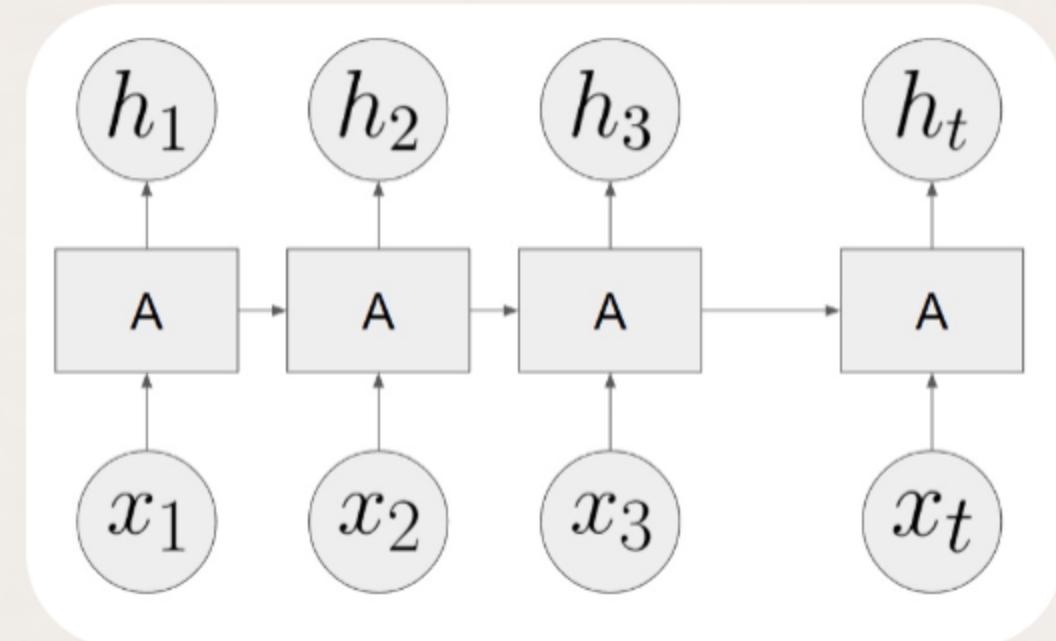
PROJECT PLAN

프로젝트 계획표



meta data

시계열 데이터



RNN

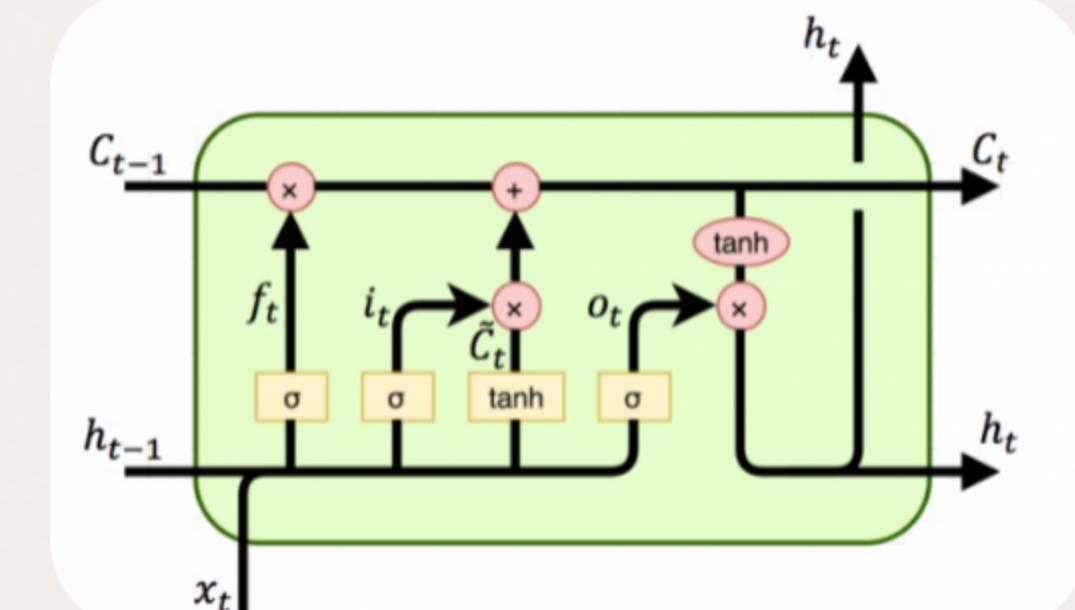
RNN 알고리즘은 순서를 가진 시퀀스 데이터(sequence data) 형태를 처리하는 것에 최적화 되어 있다. 입력 데이터는 문장이나 소리, 시계열 데이터와 같은 순서를 가지고 있는 형태의 데이터다.

인공신경망 알고리즘 A를 통하여 나온 결과값이 이며 이 값은 그 다음 연산에도 영향을 주어 출력 결과들이 이전의 결과값에 지속적으로 영향을 미친다.

LSTM

RNN은 비교적 짧은 시퀀스 데이터에서 효과적인데 이는 장기 의존성 문제(the problem of Long-Term Dependencies)를 가져 이전의 정보가 마지막까지 충분히 전달되지 않는 것으로 LSTM(Long Short Term Memory) 알고리즘을 사용하여 해결하였다.

LSTM은 RNN을 발전시킨 것으로 데이터를 계산하는 과정에서 각 상태값을 조작하는 입력, 망각, 출력 게이트를 사용하여 불필요한 연산, 오차 등을 줄였다. LSTM은 긴 시퀀스 데이터의 입력 값에서 대표적인 자질을 생성하는 능력이 우수하다.

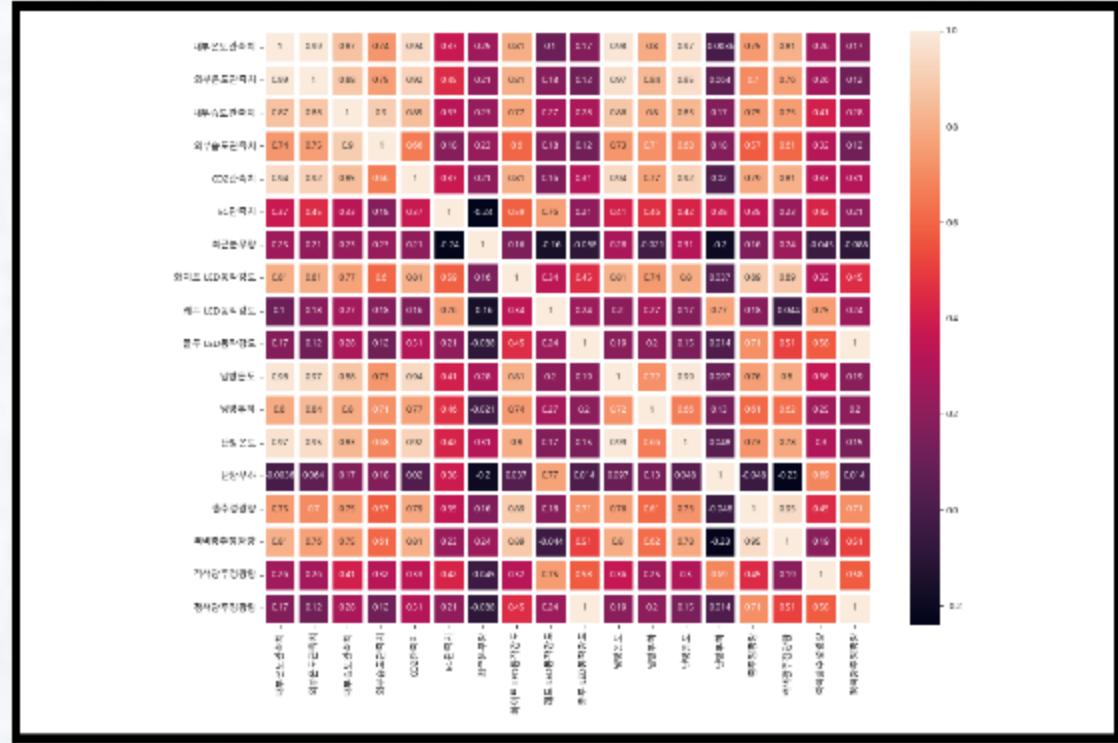


meta data

Data Pre-processing

데이터 전처리

- Meta Data 결측치 처리
 - ▼ 세부내용
 - CASE02_10, 11 → 빈 데이터
 - CASE05_22, 23 → 최근분무량 642997 이상치 이슈
 - CASE08_1 ~ 11 (전체) → 빈 데이터
 - CASE09_1 ~ 9 (전체) → 빈 데이터
 - CASE12_42 → row 902 ~ 1440 빈 데이터 → 채우지 않고 삭제
 - CASE15_1, 2 → 내부온도, 습도, 냉방, 난방부하 결측치가 많음 → `ffill`로 채움 → mean값으로 채우는게 더 좋은 결과가 나을수도 있음
 - CASE18_1, 11 → 내부온도, 습도, 냉방, 난방부하 결측치가 많음 → `ffill`로 채움 → mean값으로 채우는게 더 좋은 결과가 나을수도 있음
 - CASE22_1 ~ 11 (전체) → 빈 데이터
 - CASE23_1 ~ 9 (전체) → 빈 데이터



데이터 전처리

Meta data 결측치 처리
세부내용
결측치 없앤 training data
준비

데이터 확인

Hit map

Hitmap을 통해 데이터의
상관관계 확인

Data loder

시계열 데이터

sliding window를 통해서 1440분
을 기준으로 데이터를 분리

Dataframe Concat

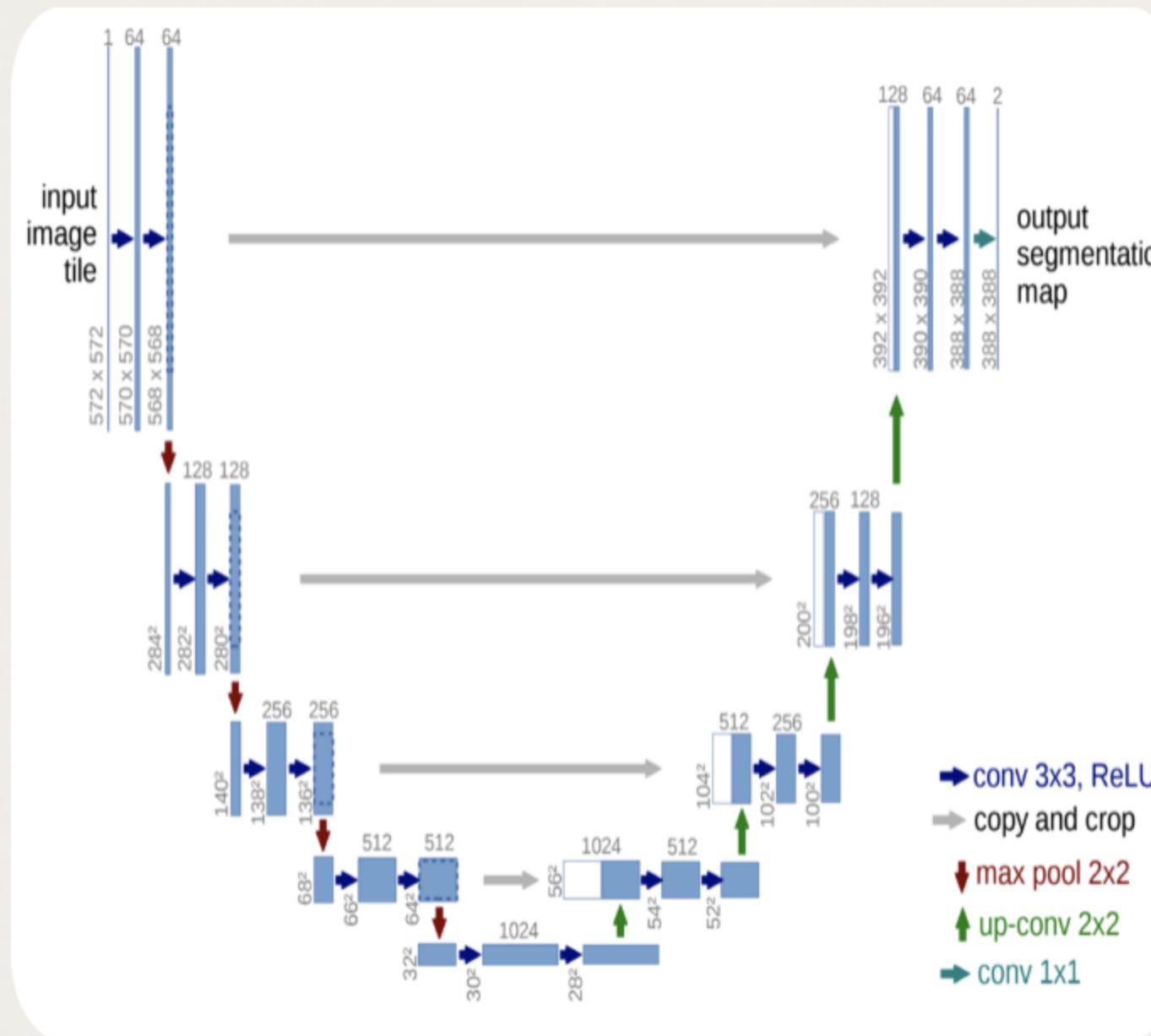
-> Convert Numpy

-> MinMax Scaler

-> Slinding Window
Processing

-> TensorDataset

image segmentation



U-Net

U-Net은 Biomedical 분야에서 이미지 분할(Image Segmentation)을 목적으로 제안된 End-to-End 방식의 Fully-Convolutional Network 기반 모델이다. 네트워크 구성의 형태('U')로 인해 U-Net이라는 이름이 붙여졌다.

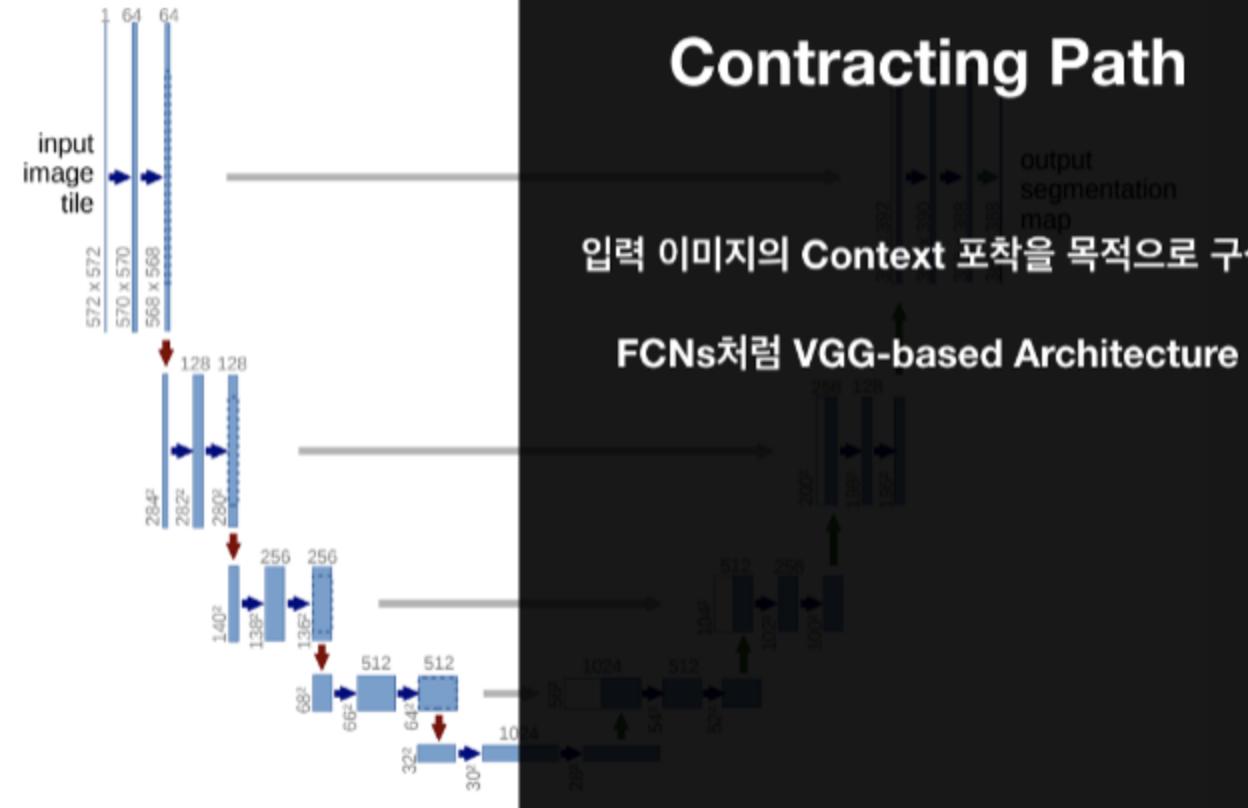
U-Net은 이미지의 전반적인 컨텍스트 정보를 얻기 위한 네트워크와 정확한 지역화(Localization)를 위한 네트워크가 대칭 형태로 구성되어 있다.

Contracting Path에서는 image의 fixture 정보를 얻기 위해 Down-sampling을 진행한다.

Expanding Path의 경우 Contracting Path의 최종 특징 맵으로부터 보다 높은 해상도의 Segmentation 결과를 얻기 위해 몇 차례의 Up-sampling을 진행한다.

image segmentation

U-Net



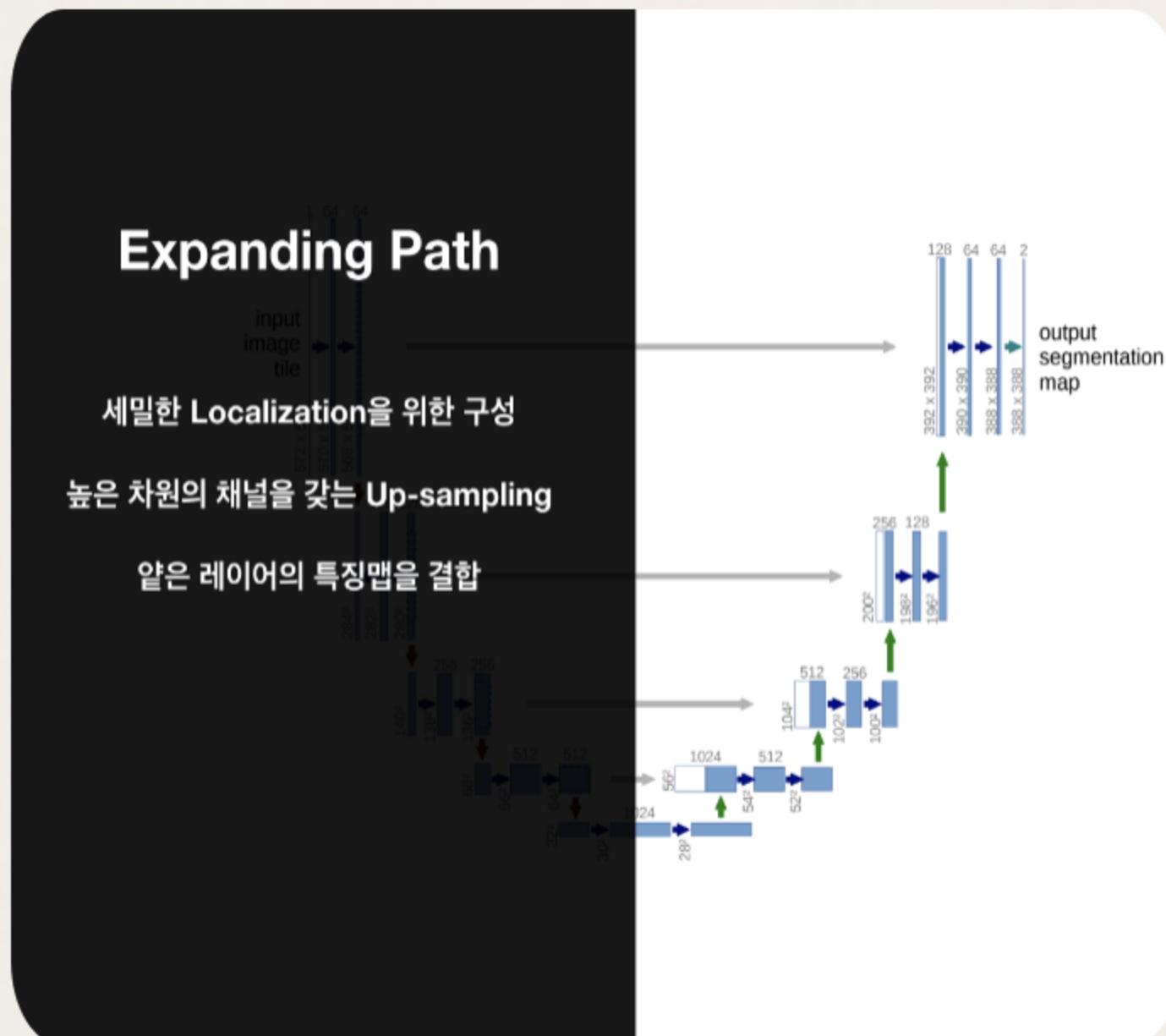
U-Net은 Biomedical 분야에서 이미지 분할(Image Segmentation)을 목적으로 제안된 End-to-End 방식의 Fully-Convolutional Network 기반 모델이다. 네트워크 구성의 형태('U')로 인해 U-Net이라는 이름이 붙여졌다.

U-Net은 이미지의 전반적인 컨텍스트 정보를 얻기 위한 네트워크와 정확한 지역화(Localization)를 위한 네트워크가 대칭 형태로 구성되어 있다.

Contracting Path에서는 image의 fixture 정보를 얻기 위해 Down-sampling을 진행한다.

Expanding Path의 경우 Contracting Path의 최종 특징 맵으로부터 보다 높은 해상도의 Segmentation 결과를 얻기 위해 몇 차례의 Up-sampling을 진행한다.

image segmentation



U-Net

U-Net은 Biomedical 분야에서 이미지 분할(Image Segmentation)을 목적으로 제안된 End-to-End 방식의 Fully-Convolutional Network 기반 모델이다. 네트워크 구성의 형태('U')로 인해 U-Net이라는 이름이 붙여졌다.

U-Net은 이미지의 전반적인 컨텍스트 정보를 얻기 위한 네트워크와 정확한 지역화(Localization)를 위한 네트워크가 대칭 형태로 구성되어 있다.

Contracting Path에서는 image의 fixture 정보를 얻기 위해 Down-sampling을 진행한다.

Expanding Path의 경우 Contracting Path의 최종 특징 맵으로부터 보다 높은 해상도의 Segmentation 결과를 얻기 위해 몇 차례의 Up-sampling을 진행한다.

image segmentation



1. Image annotation

image masking

VGG Image Annotator사이트
를 이용하여 이미지 masking

```
import albumentations as A
import cv2

img = cv2.imread("./data/segmentation/train/CASE01_04.png")
mask = cv2.imread("./data/segmentation/trainannot/CASE01_04.png")

augmentation = A.Compose([
    A.Resize(320, 320),
    A.RandomCrop(width = 128, height=128),
    A.HorizontalFlip(p=0.3)
])

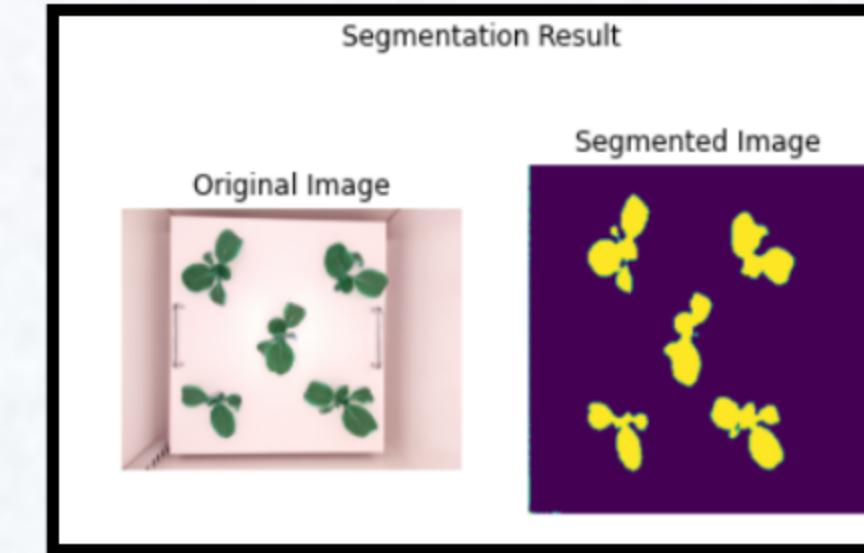
transformed = augmentation(image=img, mask=mask)
## transformed는 dictionary 형태로 img와 mask를 출력
t_img = transformed['image']
t_mask = transformed['mask']

test = [t_img, t_mask[..., 0].squeeze()]
fig = plt.figure()
for i in range(2):
    ax = fig.add_subplot(1,2,i+1)
    ax.imshow(test[i])
    ax.axis('off')
plt.show()
```

2. Albumentation

image augmentation

원본 이미지와 마스킹 이미지를
동일하게 augmentation하는
외부 library를 사용



3. result

Image Segmentation

U-Net 모델을 이용하여 Segmentation
결과 다음과 같이 원본이미지와 마스킹 된
이미지를 확인

```
def calc_pixels_from_mask(masks_dir, masks) :
    pixels = []

    for mask in masks :
        path = os.path.join(masks_dir, mask)
        img = cv2.imread(path)

        # non zero pixel 수 계산
        pix = np.count_nonzero(img)
        pixels.append(pix)

    return pixels
```

4. linear regression

code

masking image에 pixel 수를
계산하여 잎 면적 예측

PRE-TRAINING

사전학습 모델

ResNet

ResNet은 기본적으로 VGG-19의 구조를 뼈대로 한다. 거기에 컨볼루션 층들을 추가해서 깊게 만든 후에, shortcut들을 추가하는 것이 사실상 전부다.



Efficient Net

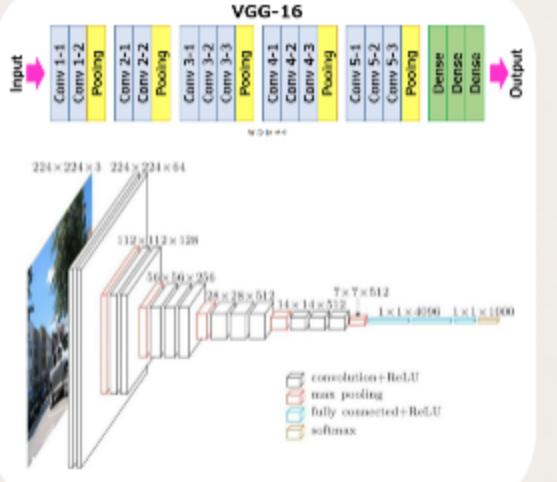
1. network의 depth를 깊게 만드는 것
2. channel width(filter 개수)를 늘리는 것(width가 넓을 수록 미세한 정보가 많이 담아짐)
3. input image의 해상도를 올리는 것

EfficientNet은 이 3가지의 최적의 조합을 실험적으로 찾았던 모델이다.



VGGNet

전체적인 구조는 AlexNet과 유사하지만 3×3 필터를 stride=1로 설정해서 반복해서 적용한다.



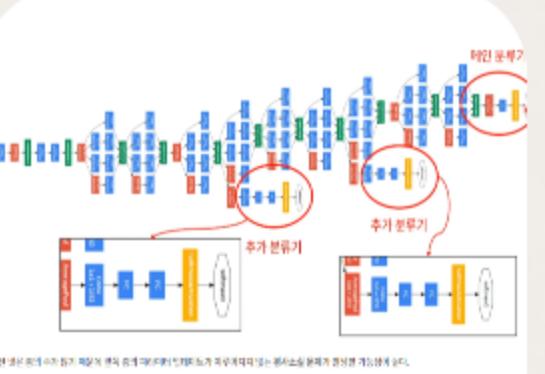
pooling layer를 제외하고 총 16개 층이어서 VGG-16이라고 부른다. pooling layer에는 2×2 로 max pooling을 stride=2로 해서 적용한다.

VGG-16과 VGG-19가 있는데 주로 16 버전이 사용된다.

Inception Net

일반 CNN 모델은 층마다 ConvNet을 한 번만 수행한다. 따라서 합성곱 필터의 크기와 풀링 레이어의 사용 위치가 모델의 성능을 구분하는 핵심 요소였다.

구글 팀에서는 이전 층에서 입력되는 값에 대해서 합성곱 필터와 풀링 필터를 포함해서 여러개의 필터를 한번에 적용했다. 이 층을 인셉션 모듈이라고 하고, 이 방식을 통해서 각 층에서 더 다양한 정보를 추출하면서 학습 속도를 개선한다.



$$\text{total_loss} = \text{real_loss} + (0.3 * \text{aux_loss_1}) + (0.3 * \text{aux_loss_2})$$

인셉션 모듈은 학습률을 적게 설정한다.

PRE-TRAINING

```

PARAMS = {
    'pretrain_net' : ['resnet50', 'inception_v2_resnet'],
    'learning rate' : [1e-2, 1e-3, 1e-4, 1e-5],
    'optimizer' : ['adam', 'rmsprop', 'nadam'],
    'dropout' : [0.2, 0.5]
}

hyper_params=list(product(PARAMS['pretrain_net'],
                           PARAMS['learning rate'],
                           PARAMS['optimizer'],
                           PARAMS['dropout']))

```

```

def bulid_model(hyper_params : tuple) :
    pretrain_net, learning_rate, opti, dropout = hyper_params

    if pretrain_net == 'resnet50' :
        MODEL_IMAGE_SIZE = 224
        base_model = resnet50.ResNet50(
            weights='imagenet',
            include_top = False,
            input_shape = (MODEL_IMAGE_SIZE, MODEL_IMAGE_SIZE,3)
        )
    elif pretrain_net == 'inception_v2_resnet':
        MODEL_IMAGE_SIZE = 299
        base_model = inception_resnet_v2.InceptionResNetV2(
            weights='imagenet',
            include_top = False,
            input_shape = (MODEL_IMAGE_SIZE, MODEL_IMAGE_SIZE,3)
        )

    base_model.trainable=False

```

```

def fit_model(model=None, early_stop = True, epoch=100, ckpt_path=None, save_best_on
              save_weights_only=True) :
    es = EarlyStopping(monitor='loss',
                       mode='auto',
                       patience=5,
                       verbose=1)

    checkpointer = ModelCheckpoint(filepath=ckpt_path,
                                   monitor='val_nmae',
                                   save_weights_only = save_weights_only,
                                   save_best_only= save_best_only,
                                   verbose=1)

    call_backs = [checkpointer]
    if early_stop :
        call_backs.append(es)

```

Hyperparameter 조정



48 Case

Hyperparameter 탐색

구현 코드 : bulid_model



code

모델 구현 코드

구현 코드 : fit_model

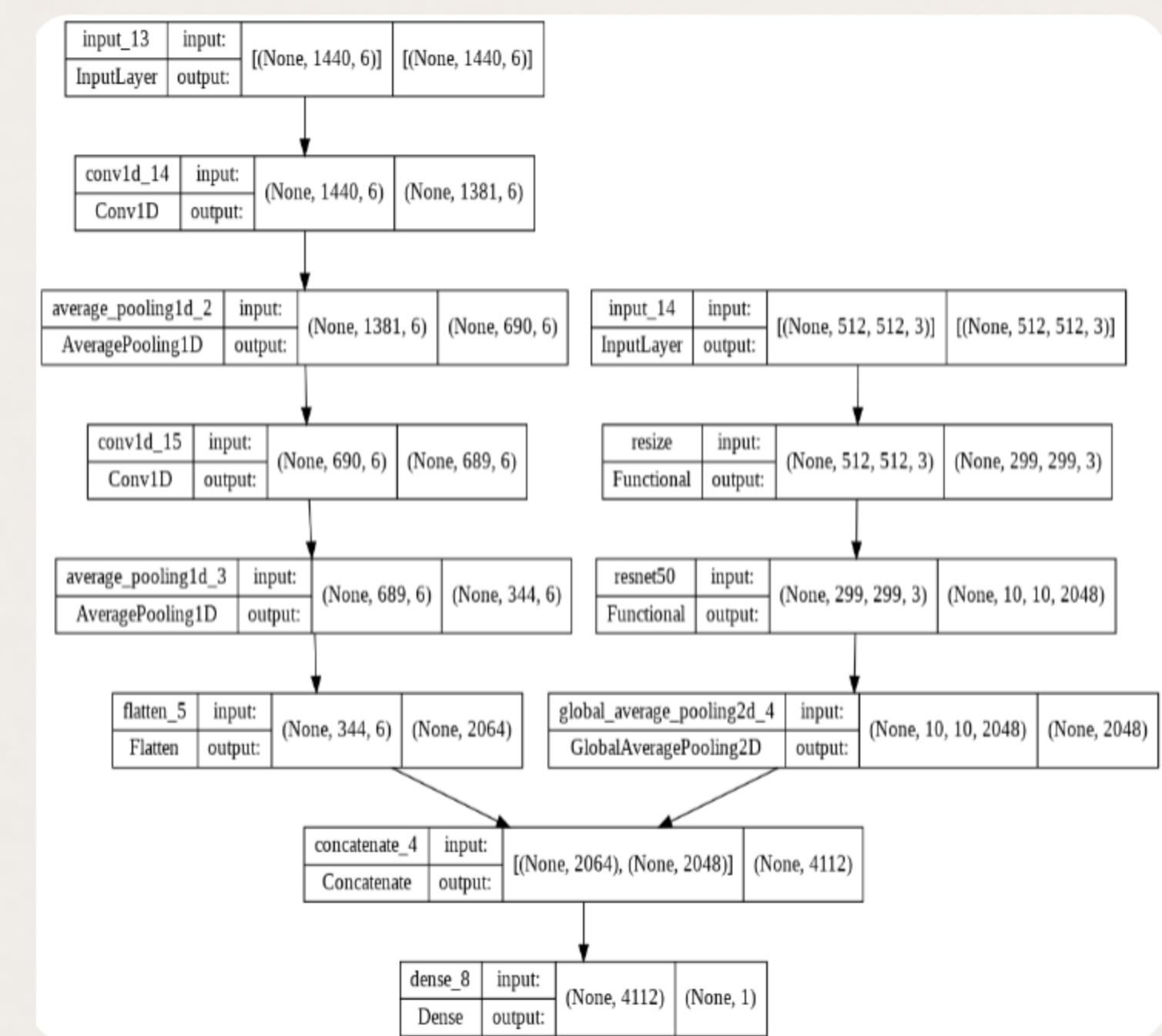


code

모델 학습 코드

MULTI INPUT MODEL

ResNet & Conv1D



다중입력 모델

데이터 특성에 따른 서로 다른 여러개의 모델이 input으로 사용되어 하나의 output을 내는 네트워크 구조이다.

05

수행결과

meta data

RNN, LSTM

RNN Model

RNN 모델은 시계열 데이터를 1시간 단위로 잘라 학습을 진행하였다.

LSTM 모델은 시계열 데이터를 1분 단위로 잘라 학습을 진행하였다.

CASE	Model	learning_rate	Optimizer
01	RNN	1e-3	rmsprop
02	LSTM	3e-3	rmsprop

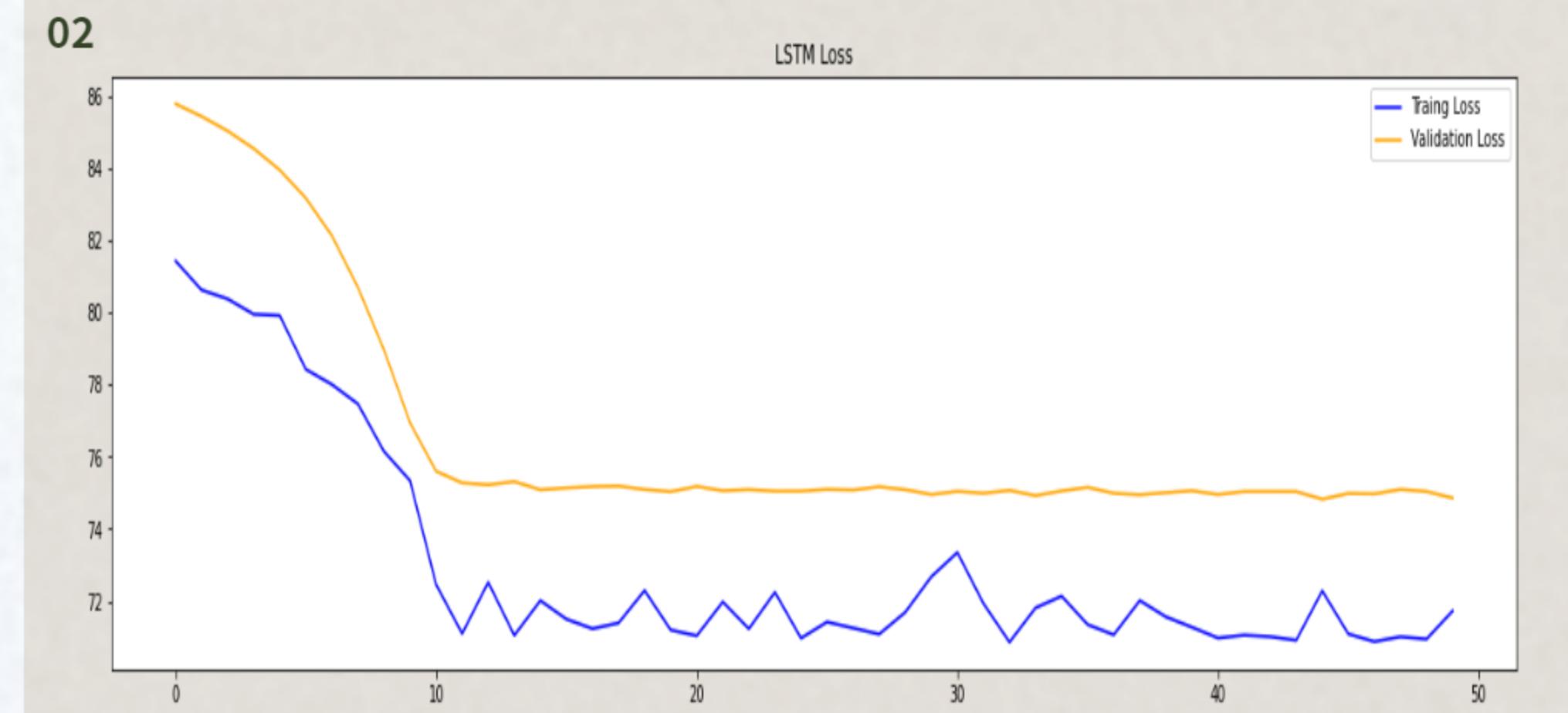
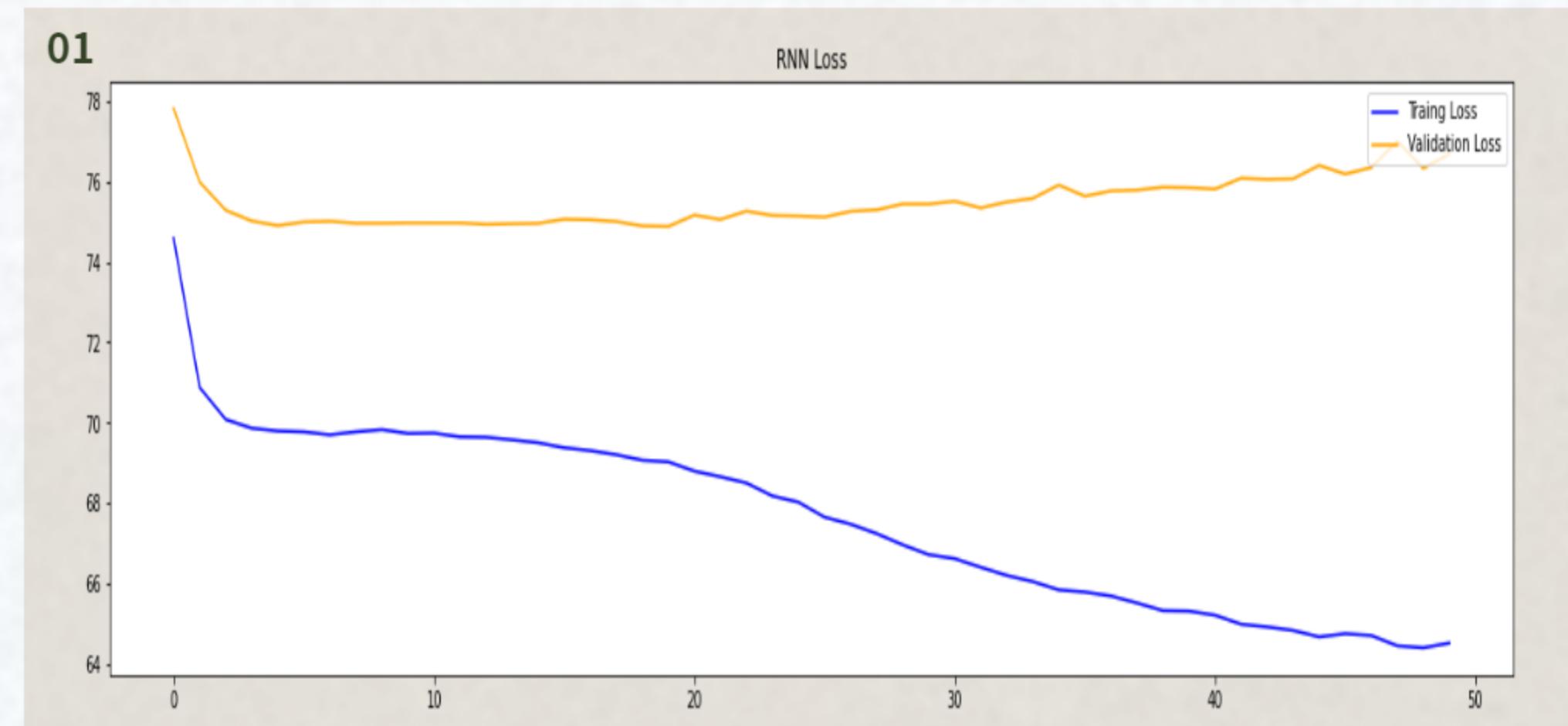


image segmentation

U-Net

CASE	learning_rate	Optimizer	epoch	augmentation
01	1e-4	adam	10	0
02	1e-4	adam	3	X

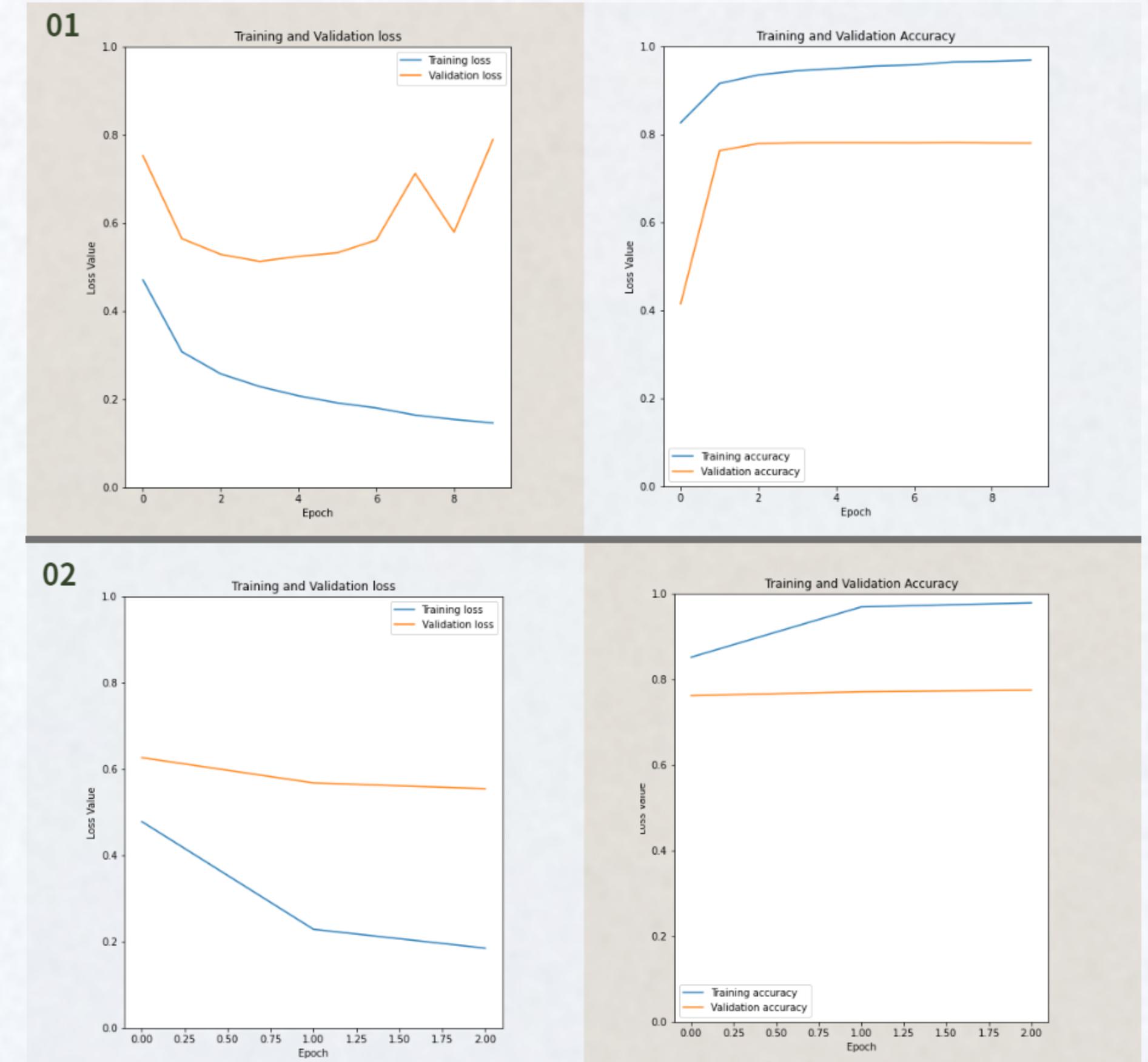
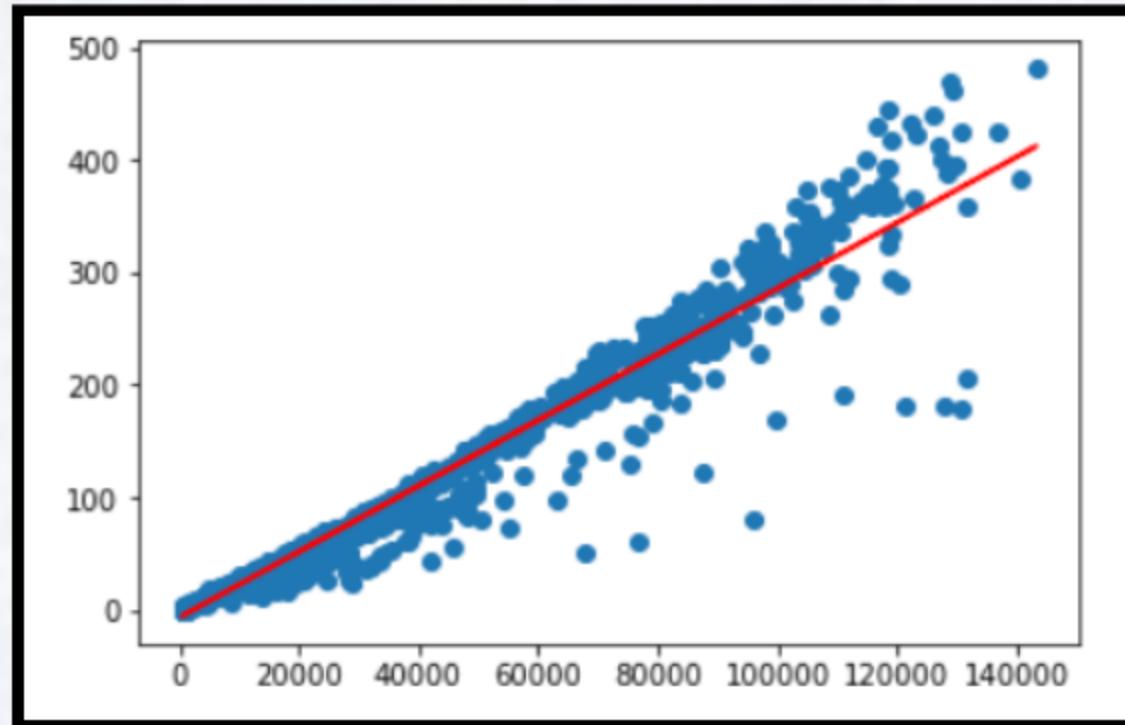


image segmentation

잎 면적 계산



잎 면적 계산

픽셀 수 계산(np.nonzero())

x축 : 픽셀
y축 : label

Tensorflow 예측 결과

잎 면적을 통한 결과

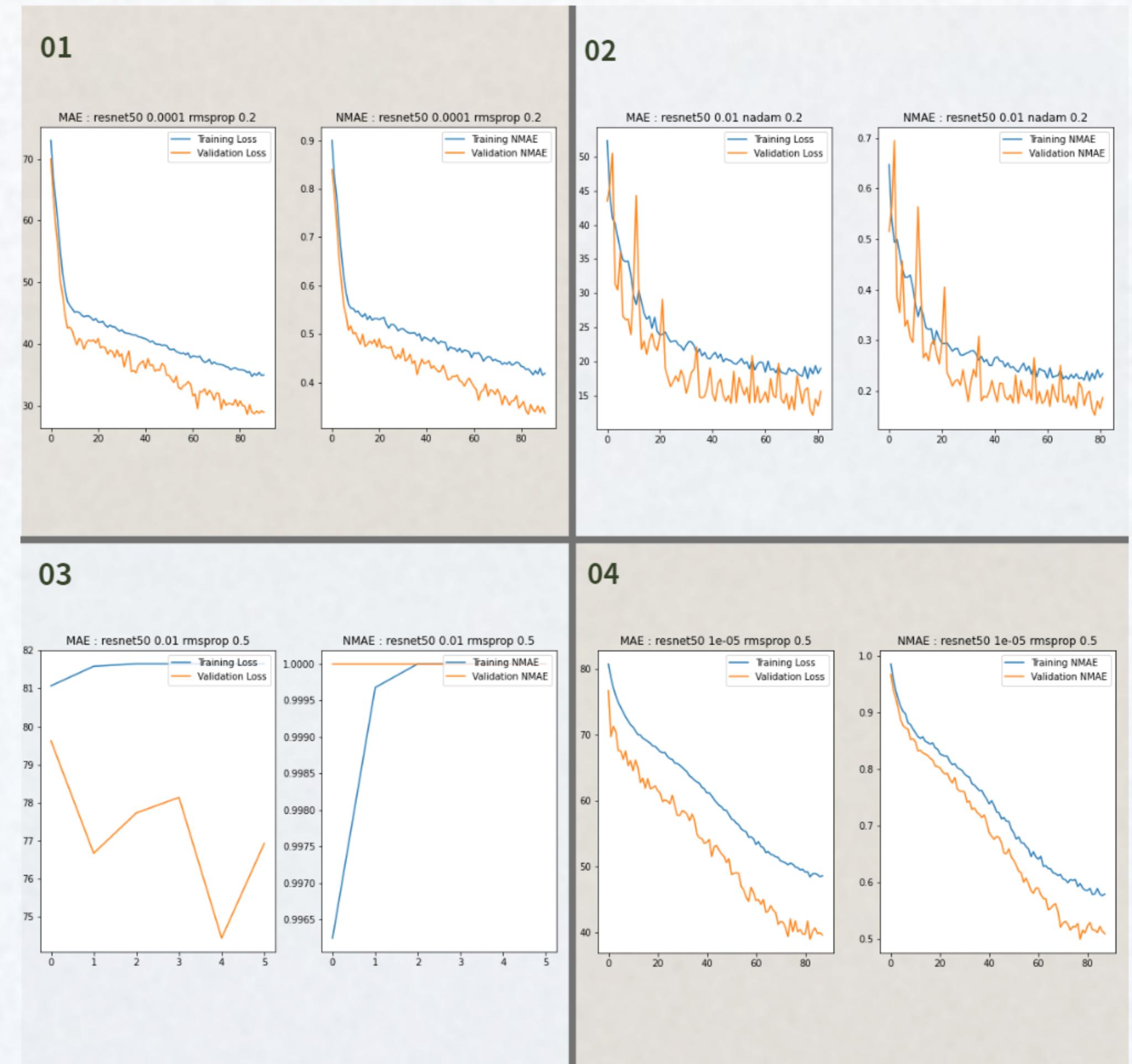
결과 값 : 0.4242113733

PRE-TRAINING

ResNet, Efficient Net,
VGGNet, inception Net

ResNet50

CASE	Pretrain_net	learning_rate	Optimizer	droup_out
01	resnet50	0.0001	rmsprop	0.2
02	resnet50	0.01	nadam	0.2
03	resnet50	0.01	rmsprop	0.5
04	resnet50	1e-05	rmsprop	0.5

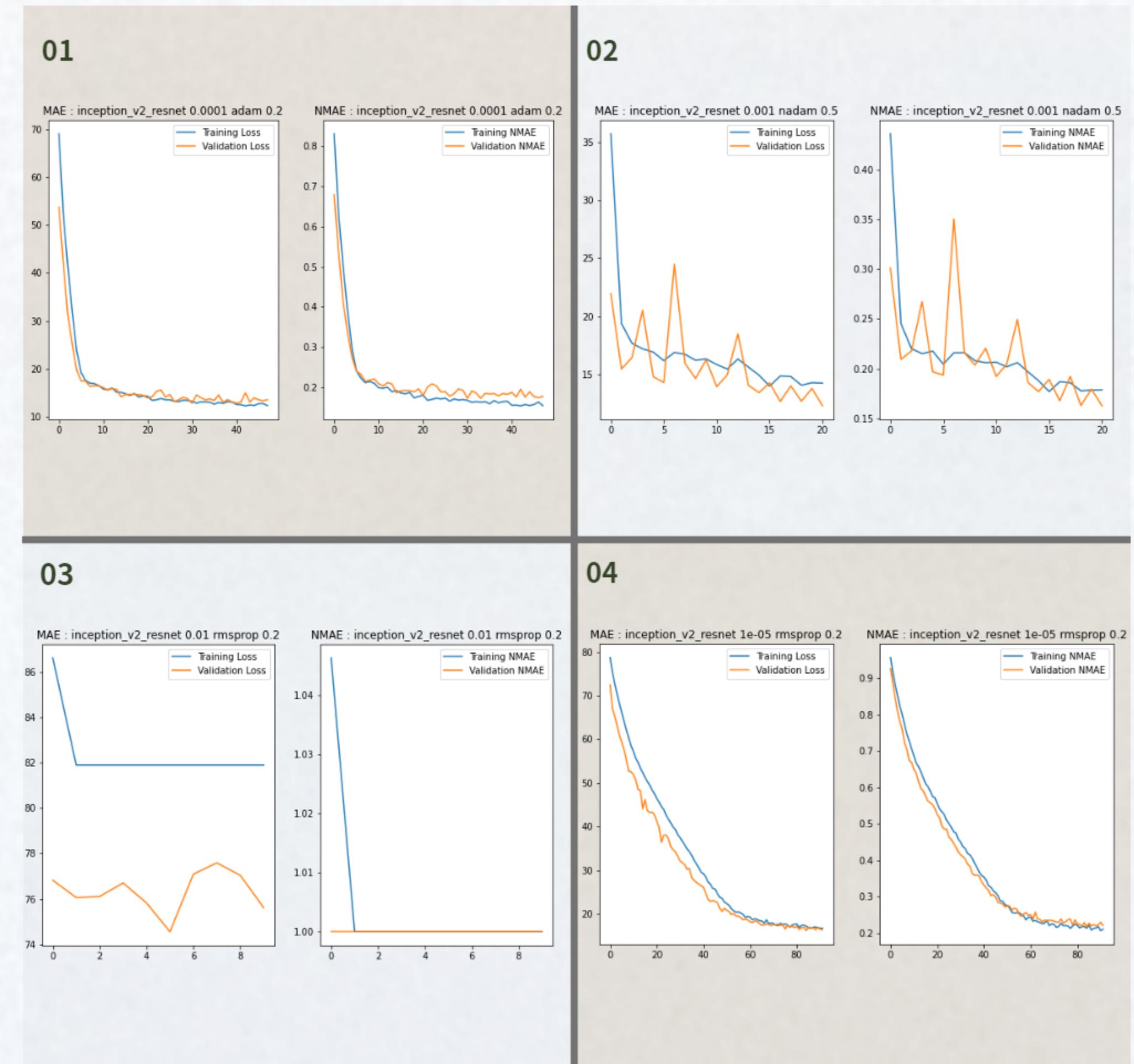


PRE-TRAINING

ResNet, Efficient Net,
VGGNet, inception Net

inception_v2_resnet

CASE	Pretrain_net	learning_rate	Optimizer	droup_out
01	inception resnetv2	0.0001	adam	0.2
02	inception resnetv2	0.001	nadam	0.5
03	inception resnetv2	0.01	rmsprop	0.2
04	inception resnetv2	1e-05	rmsprop	0.2



Fine tuning

InceptionResNetV2

Learning Rate : 1e-3

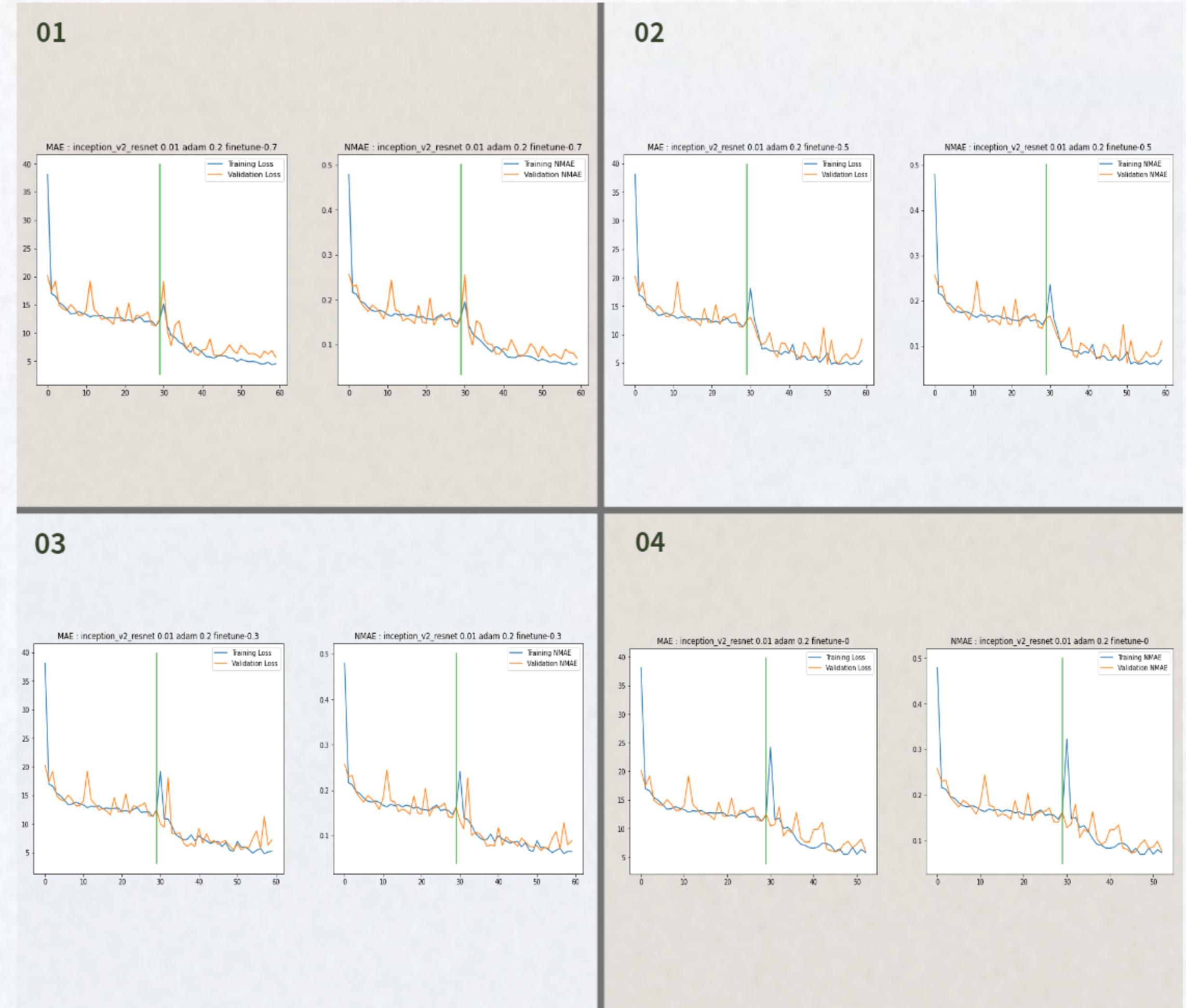
Optimizer : Adam

Dropout : 0.2

Freezing Rate

finetuning시 데이터의 종류, 양에 따라서
pretrain net의 동결 비율을 정함

CASE	Freezing
01	0.7
02	0.5
03	0.3
04	0



RESULT

Pretrain & Finetuning

Pretrain & Finetuning Results			
inception v2 resnet 1e-3 adam 0.2 finetune 0.csv	2022-05-17 23:46:31	0.1980762807	<input checked="" type="checkbox"/>
inception v2 resnet 1e-3 adam 0.2 finetune 0.3.csv	2022-05-17 23:15:26	0.2102609314	<input type="checkbox"/>
inception v2 resnet 1e-3 adam 0.2 finetune 0.5.csv	2022-05-17 22:34:51	0.2150525839	<input type="checkbox"/>
inception v2 resnet 1e-3 adam 0.2 finetune 0.7.csv	2022-05-17 21:44:44	0.2484155641	<input type="checkbox"/>
inception v2 resnet 1e-3 adam 0.2.csv	2022-05-17 21:03:00	0.295630288	<input type="checkbox"/>

MULTI INPUT MODEL

pytorch

```

for epoch in range(num_epochs):
    model.train()
    train_loss = []
    for img_data, meta_data in tqdm(zip(train_loader, train_meta_dataloader)):

        x_train, label = img_data
        x_meta_train, meta_label = meta_data

        optimizer.zero_grad()
        outputs = model(x_train.to(device), x_meta_train.to(device))
        out = torch.squeeze(outputs)

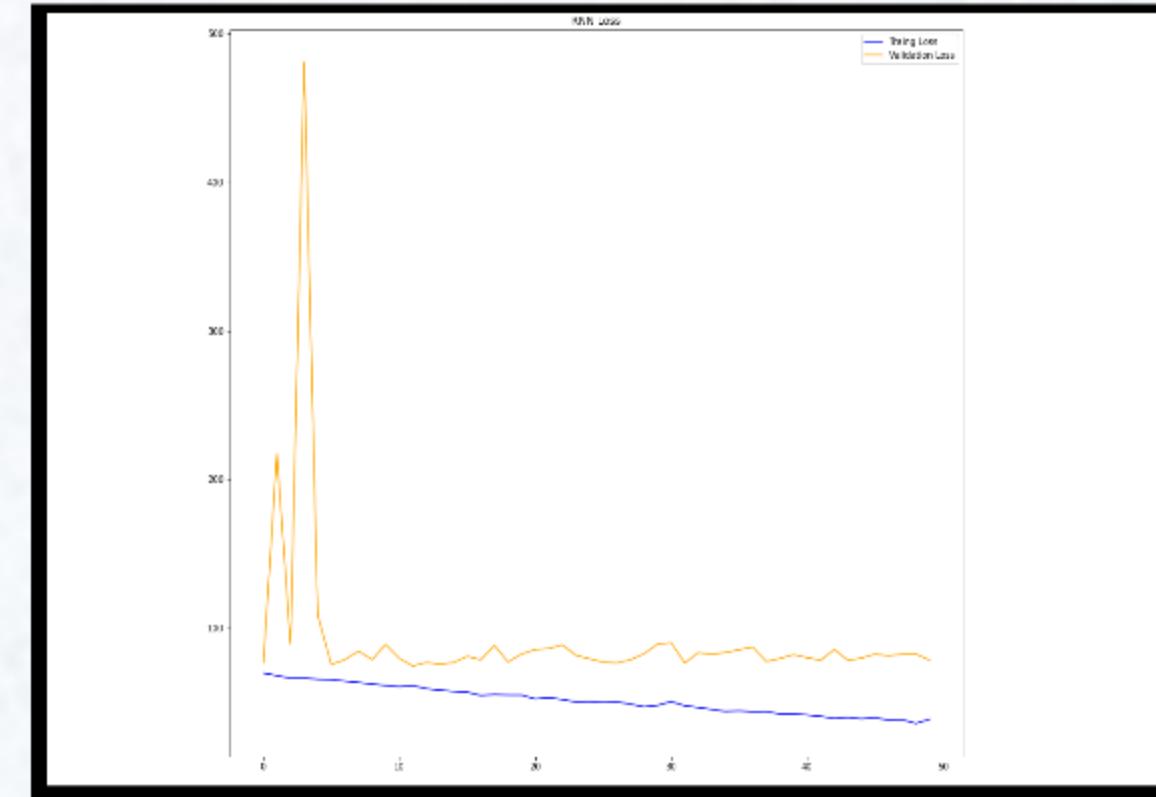
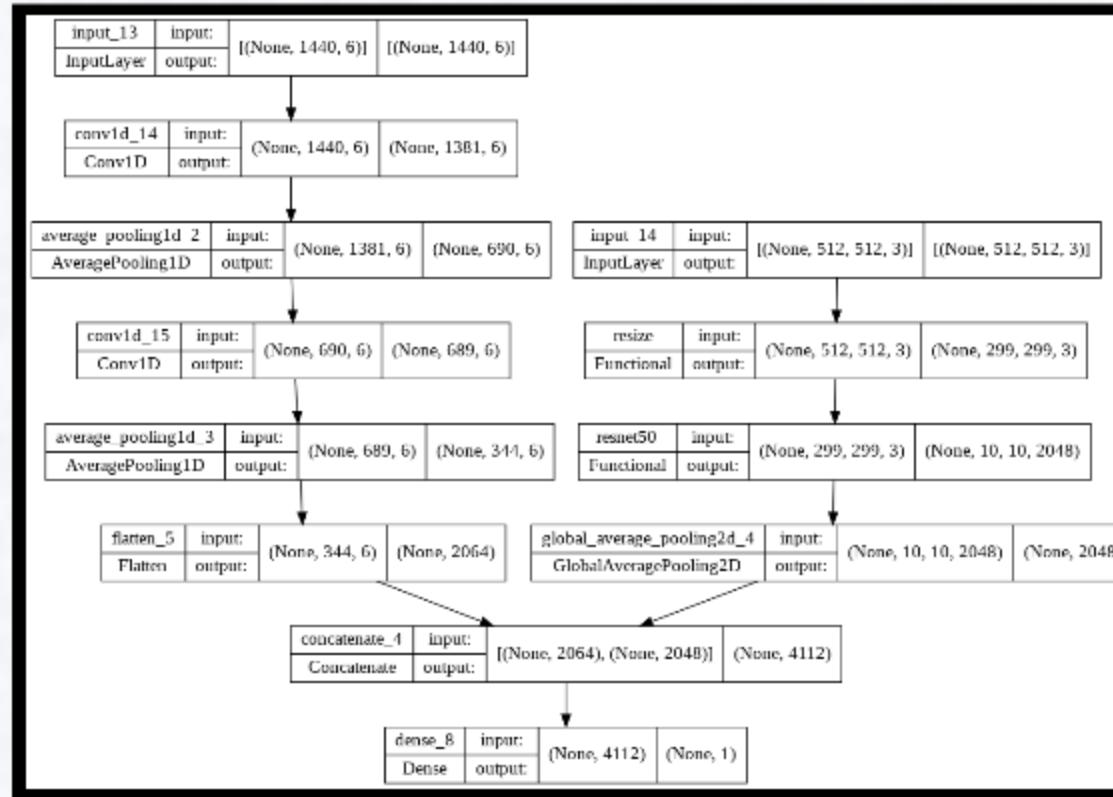
        # obtain the loss function
        loss = criterion(out, label.to(device))

        loss.backward()

        optimizer.step()

    train_loss.append(loss.item())

```



DataLoader

(img data, meta data)

전처리 완료된 이미지와 메타데이터
이터를 같은 배치사이즈로
DataPipeline 구축

Data model

multi input model

multi input model의
모델 구조

result

loss

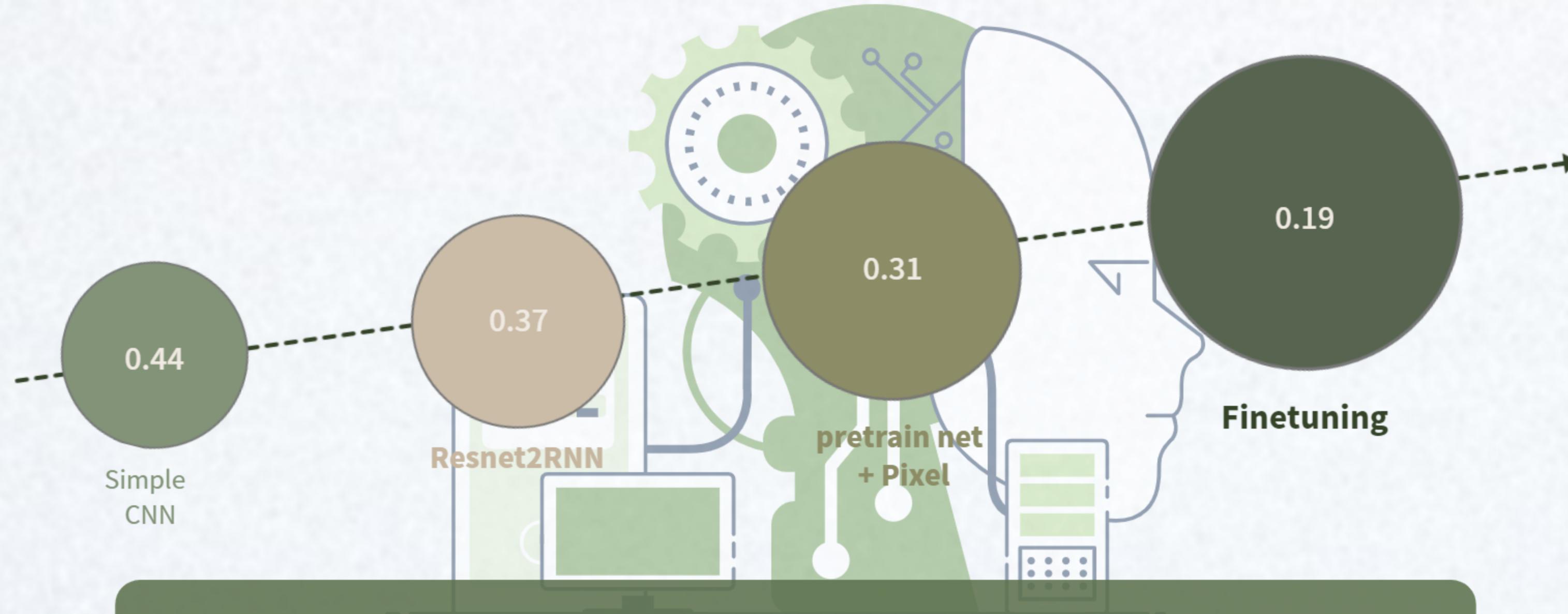
Training
Validation
loss graph

06

결론

PERFORMANCE

보완 과정



프로젝트 최종 결과

meta data를 이용하지 않고 이미지만 이용한 모델에서 NMAE값 0.19를 얻을 수 있었음. 시계열 데이터를 적절히 전처리 및 이용하였으면 더 좋은 결과를 얻을 수 있었을 것

PROBLEMS

복잡한 데이터 구조

이미지와 메타데이터 연결
의 어려움

적합한 모델 찾기

두가지 데이터를 동시에 처리 할 수 있는 모델 선정이 어려움

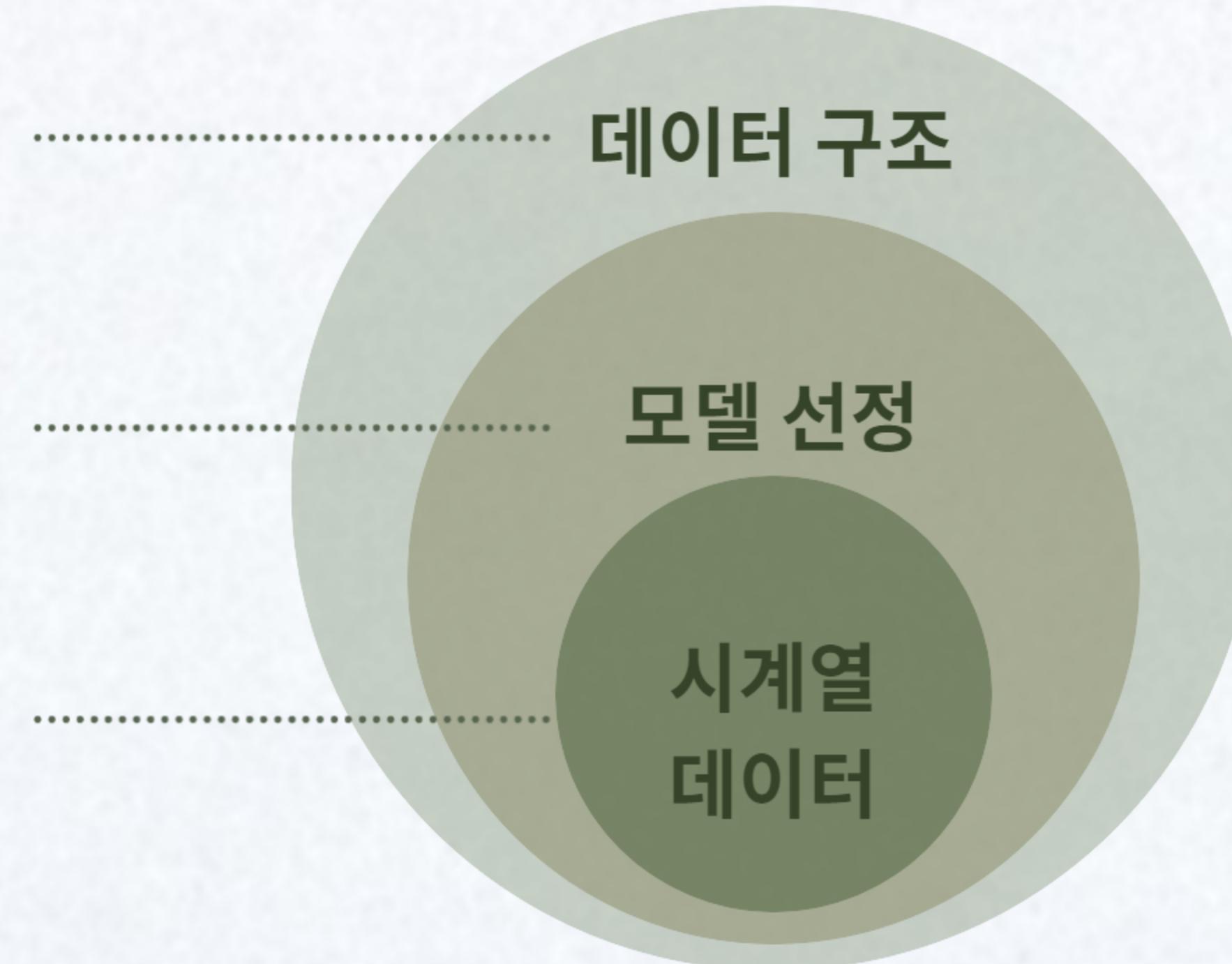
모델 최적화 문제

시계열 데이터가 포함된 모델은 최적화가 힘듦

데이터 구조

모델 선정

시계열 데이터



Feedback

김인후

시계열 데이터와 이미지 데이터의 조화가 처음부터 끝까지 지속적으로 문제가 되었다. 시계열데이터만 사용하면 성능이 하락하였으며 명확한 참고자료가 없어서 힘들었다.

이희경

데이터의 중요성을 느끼고 여러가지 모델들을 적용해볼 수 있는 기회였다. 시계열데이터에 대해 더 알아볼 수 있는 시간이 충분하지 않아서 아쉽다.

유영재

이미지 세그멘테이션을 진행하면서 난항을 많이 겪었다. 모르는 부분이 많았는데 조원들의 도움으로 좋은 결과도 얻을 수 있었다. 여러 모델을 사용할 수 있는 좋은 경험이였다.

장소희

학습 모델도 매우 중요했지만 사전데이터도 생각했던것보다 중요하다는게 크게 와닿았다 혼자서 막히는 부분도 많았는데 팀원들이 잘 이끌어줘서 생각보다 좋은 결과를 낼 수 있었다

총평

이번 프로젝트는 최대한 대면으로 진행하여 커뮤니케이션이 원활하게 진행되었고, 많은 작업을 수행할 수 있었다. 하지만 시계열 데이터가 마지막까지 문제를 줘서 좋은 결과를 얻지 못한점이 아쉽다.



THank YOU

