고려대학교 세종캠퍼스
KOREA UNIVERSITY SEJONG CAMPUS

병충해 진단 시스템

팀명 : 코끼리(코딩 하는 사람들 끼리끼리)

팀원: 장미훈(팀장), 김도형(발표자), 기회석, 장해섭

고려대학교 세종캠퍼스
KOREA UNIVERSITY SEJONG CAMPUS

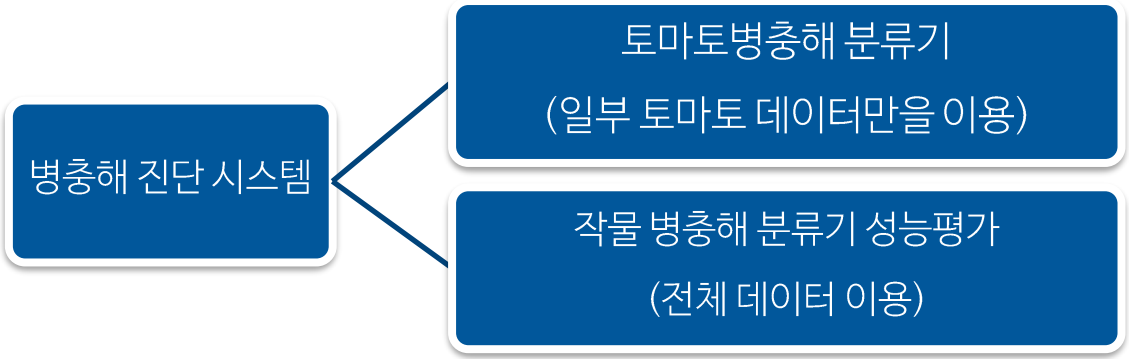
목차

1. 분석목적

- 도시농업과 도시근로자 은퇴 후 귀농과 귀촌, 젊은 농업인과 스마트 농업 등이 이슈화되고 있는 요즈음, 처음부터 병충해를 미리 예방하거나 방제하기는 쉽지 않다.
- 작물이미지 정보를 이용한 AI병충해분류기를 만들어 작물병충해를 분류하고 진단할 수 있는 시스템을 구축한다.
- 진단시스템을 구축을 함으로써 정확한 조치 수단을 최대한 신속하게 적용해 질병에 대한 작물 피해를 최소화하기 위해 프로젝트 진행

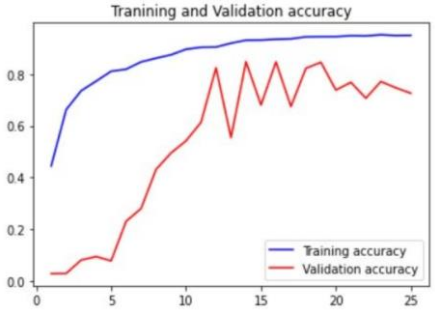
2. 분석방법

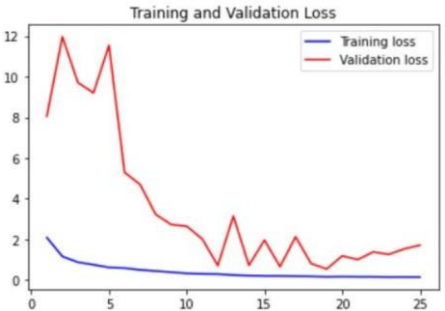
- 언어 : Python
- 패키지 : Tensorflow, Keras, NumPy, Matplotlib
- 툴 : Colab, DiCia, Jupyter Notebook

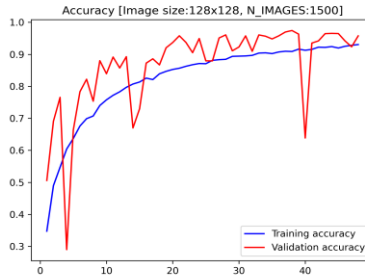


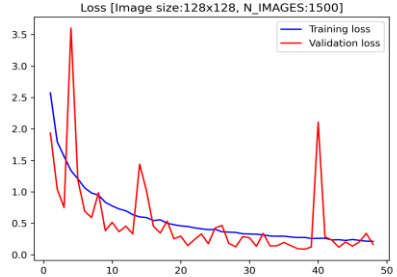
2. 분석방법_작물 병충해 분류기 성능평가(1안 ~ 6안)

구분	이미지 사이즈	이미지 수	Batch Size	Epochs	Steps	LR	Depth
1안	256*256	300	128	25	100	0,001	3
6안	128*128	1,500	8	50	100	0,001	3







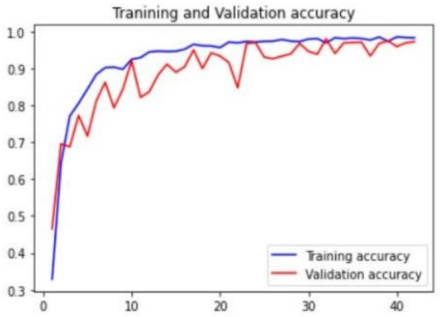


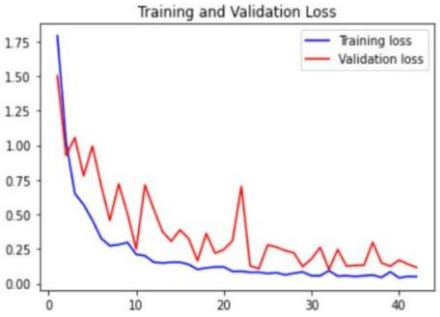
- 작물 병충해 1안에서 6안까지 이미지사이즈, Batch_size, Epochs, Steps 등 parameter를 다방면으로 변경했으나 매번 모델평가부분에서 Overfitting이 발생
- 이후 논문과 일부 데이터를 사용한 토마토분류기를 통해 새로운 방법론을 연구

2. 분석방법_토마토병충해분류기 성능평가

구분	이미지_size	이미지수 (N_IMAGE)	Batch_size	Epochs	Steps	LR	Depth	생성모델	accuracy	precision (weighted_avg)	recall (weighted_avg)	f1-score (weighted_avg)
1안	128*128	200	32	25	100	0.001	3	best_my_tomato12825_model.h5	0.91	0.91	0.90	0.90
2안	128*128	300	16	25	100	0.001	3	best_my_tomato12830_model.h5	0.9	0.9	0.89	0.89
3안	256*256	300	32	30	100	0.001	3	best_my_tomato30_model.h5	0.8	0.87	0.87	0.87
4안	256*256	500	32	50	50	0.001	3	best_my_tomato30_model.h5	0.9	0.91	0.90	0.91
5안	256*256	1500	64	100	50	0.001	3	best_my_tomato1500_model.h5	0.98	0.98	0.98	0.98

- 토마토 데이터 중 최소 데이터가 200개인점을 감안해 이미지 수를 200개로 설정하고 모델 평가 -〉
모델 평가 그래프에서 Overfitting이 없는 걸 확인
- 1안과 2 ~ 4안을 통해 균등 및 불균등한 데이터를 비교를 함으로써 균등한 데이터를 쓰는 방향으로 가닥
- 5안에서는 처음부터 데이터 가중한 데이터를 사용함으로써 accuracy, f1-score 수치가 높은걸 확인 후 작물 병충해 분류기 성능평가에 같은 방법론 적용





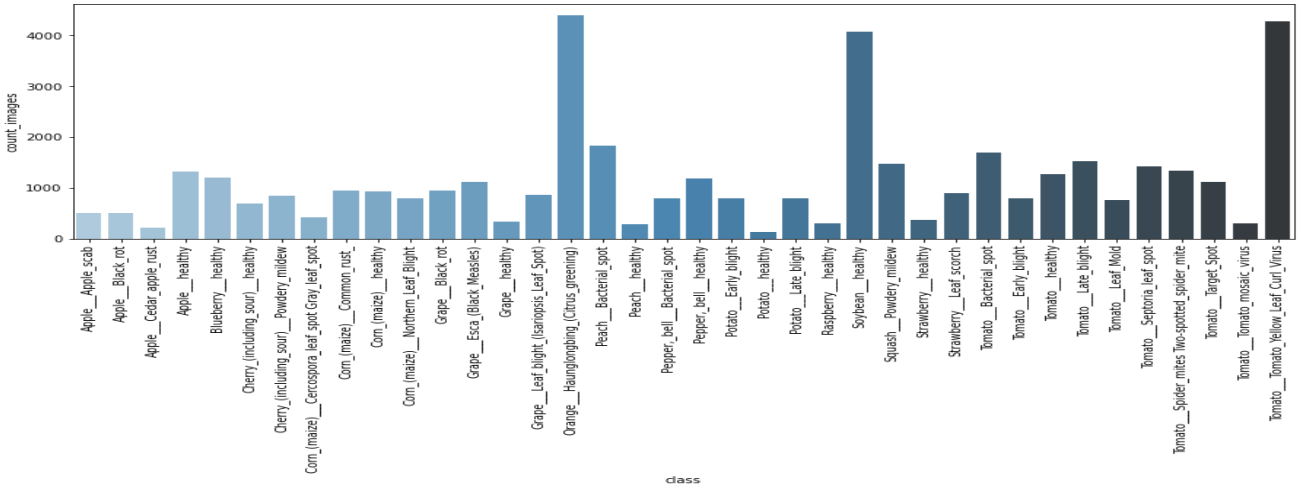
2. 분석방법_작물 병충해 분류기 성능평가(1안 ~ 8안)

구분	이미지_size	이미지수 (N_IMAGE)	Batch_size	Epochs	Steps	LR	Depth	생성모델	accuracy	precision (weighted_avg)	recall (weighted_avg)	f1-score (weighted_avg)
1안	256*256	300	128	25	100	0.001	3	best_my_model.h5	0.85	0.87	0.85	0.84
2안	128*128	300	32	25	100	0.001	3	my_model_image128x128_n300_batch32.h5	0.87	0.89	0.87	0.87
3안	128*128	500	8	25	100	0.001	3	my_model_image128x128_n500_batch8.h5	0.95	0.95	0.95	0.95
4안	128*128	1000	8	25	100	0.001	3	my_model_image128x128_n1000_batch8.h5	0.95	0.95	0.95	0.95
5안	128*128	1500	8	25	100	0.001	3	my_model_image128x128_n1500_batch8.h5	0.95	0.96	0.95	0.95
6안	128*128	1500	8	50	100	0.001	3	my_model_image128x128_n1500_batch8_epoch50.h5	0.97	0.97	0.97	0.97
7안	128*128	1500	64	50	50	0.001	3	plant_disease_aug1500_best_my_model_50.h5	0.98	0.99	0.98	0.98
8안	128*128	1600	64	50	100	0.001	3	plant_disease_aug1600_best_my_model_100.h5	0.99	0.99	0.99	0.99

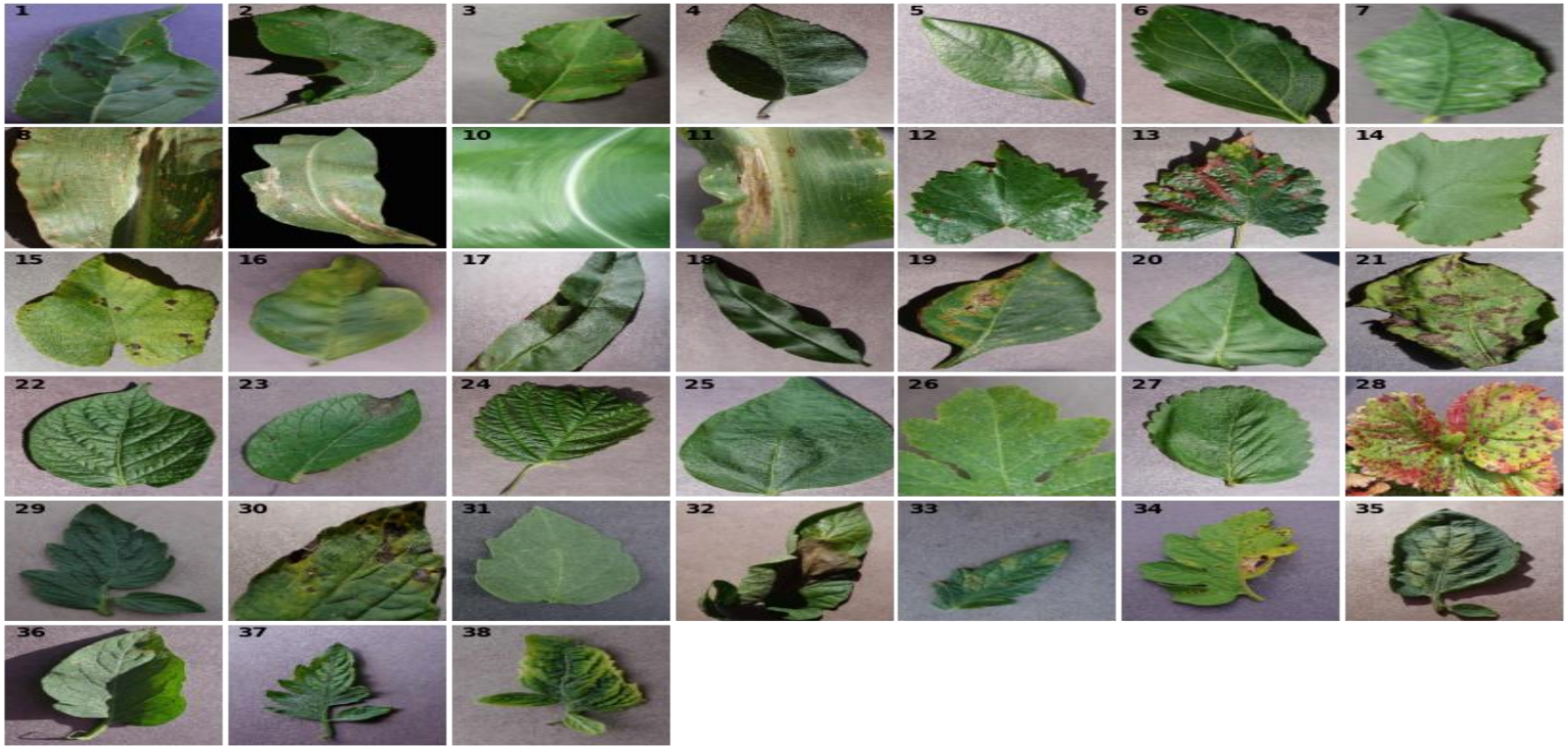
- 7안부터 처음부터 증강된 데이터를 사용해 모델 훈련
- 최종적으로 accuracy(0.99)가 높은 8안을 채택

3. 분석 데이터

PlantVillage 데이터셋은 54,303개의 건강한 잎 이미지와 건강하지 않은 잎 이미지로 구성되어 있으며 종과 질병에 따라 38개의 카테고리로 정리



PlantVillage 데이터셋 카테고리별 작물 이미지



3. 분석 데이터

	disease	count_images
0	Apple___Apple_scab	2016
1	Apple___Black_rot	1987
2	Apple___Cedar_apple_rust	1760
3	Apple___healthy	2008
4	Blueberry___healthy	1816
5	Cherry_(including_sour)___healthy	1826
6	Cherry_(including_sour)___Powdery_mildew	1683
7	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_...	1642
8	Corn_(maize)___Common_rust_	1907
9	Corn_(maize)___healthy	1859
10	Corn_(maize)___Northern_Leaf_Blight	1908
11	Grape___Black_rot	1888
12	Grape___Esca_(Black_Measles)	1920
13	Grape___healthy	1692
14	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1722
15	Orange___Haunglongbing_(Citrus_greening)	2010
16	Peach___Bacterial_spot	1838
17	Peach___healthy	1728
18	Pepper,_bell___Bacterial_spot	1913
19	Pepper,_bell___healthy	1988
20	Potato___Early_blight	1939
21	Potato___healthy	1824
22	Potato___Late_blight	1939
23	Raspberry___healthy	1781
24	Soybean___healthy	2022
25	Squash___Powdery_mildew	1736
26	Strawberry___healthy	1824
27	Strawberry___Leaf_scorch	1774
28	Tomato___Bacterial_spot	1702
29	Tomato___Early_blight	1920
30	Tomato___healthy	1926
31	Tomato___Late_blight	1851
32	Tomato___Leaf_Mold	1882
33	Tomato___Septoria_leaf_spot	1745
34	Tomato___Spider_mites Two-spotted_spider_mite	1741
35	Tomato___Target_Spot	1827
36	Tomato___Tomato_mosaic_virus	1790
37	Tomato___Tomato_Yellow_Leaf_Curl_Virus	1961

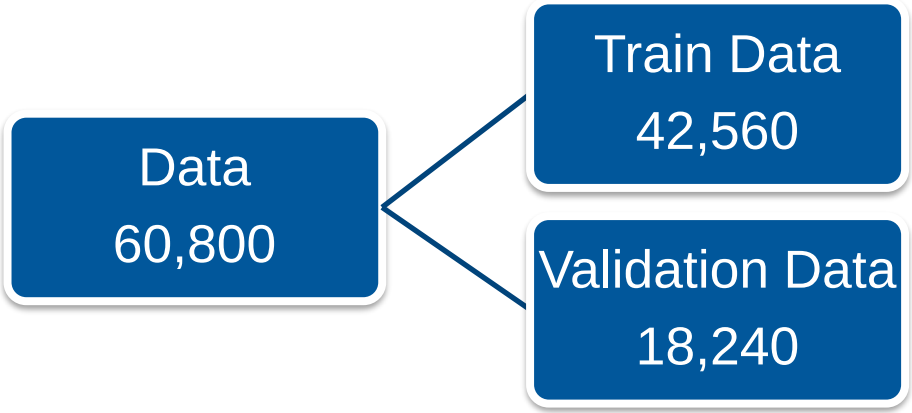
- 처음 증강된 데이터를 사용해 기존 54,303개에서 70,295개인 데이터를 사용

3. 분석 데이터

	disease	count_images
0	Apple___Apple_scab	1600
1	Apple___Black_rot	1600
2	Apple___Cedar_apple_rust	1600
3	Apple___healthy	1600
4	Blueberry___healthy	1600
5	Cherry_(including_sour)___healthy	1600
6	Cherry_(including_sour)___Powdery_mildew	1600
7	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_...	1600
8	Corn_(maize)___Common_rust_	1600
9	Corn_(maize)___healthy	1600
10	Corn_(maize)___Northern_Leaf_Blight	1600
11	Grape___Black_rot	1600
12	Grape___Esca_(Black_Measles)	1600
13	Grape___healthy	1600
14	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1600
15	Orange___Haunglongbing_(Citrus_greening)	1600
16	Peach___Bacterial_spot	1600
17	Peach___healthy	1600
18	Pepper,_bell___Bacterial_spot	1600
19	Pepper,_bell___healthy	1600
20	Potato___Early_blight	1600
21	Potato___healthy	1600
22	Potato___Late_blight	1600
23	Raspberry___healthy	1600
24	Soybean___healthy	1600
25	Squash___Powdery_mildew	1600
26	Strawberry___healthy	1600
27	Strawberry___Leaf_scorch	1600
28	Tomato___Bacterial_spot	1600
29	Tomato___Early_blight	1600
30	Tomato___healthy	1600
31	Tomato___Late_blight	1600
32	Tomato___Leaf_Mold	1600
33	Tomato___Septoria_leaf_spot	1600
34	Tomato___Spider_mites Two-spotted_spider_mite	1600
35	Tomato___Target_Spot	1600
36	Tomato___Tomato_mosaic_virus	1600
37	Tomato___Tomato_Yellow_Leaf_Curl_Virus	1600

- 그 후 각 범주들의 데이터를 각각 1,600개로 설정함으로써 정확도 향상 기대

3. 분석 데이터



- 총 데이터는 60,800개 수집
- Train / Validation data는 0.7 / 0.3 비율로 설정

4. 모델설명

- 처음 구성한 모델과는 BatchNormalization, Dropout를 삭제를 함으로써 모델을 간소화

- 7안에서 정확도 상승의 효과를 통해 8안에서도 모델 그대로 적용

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 128, 128, 32)      896
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 256)       295168
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 8, 8, 512)         1180160
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              8389632
_________________________________________________________________
dense_3 (Dense)              (None, 38)                38950
=================================================================
Total params: 9,997,158
Trainable params: 9,997,158
Non-trainable params: 0
_________________________________________________________________
```

4. 모델설명

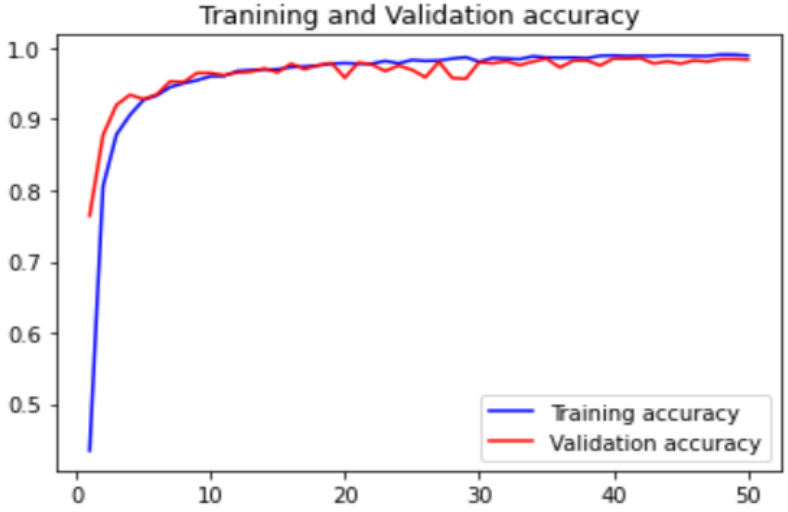
```
# Model Run

METRICS = ['accuracy',
           tensorflow.keras.metrics.Precision(name='precision'),
           tensorflow.keras.metrics.Recall(name='recall') ]

# Initialize optimizer
opt = Adam(learning_rate=LR, decay=LR / EPOCHS)

# Compile model
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=METRICS)

# Train model
print('[INFO] Training network ...')
callbacks = [tensorflow.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.005, patience=5, min_lr=0.001),
             tensorflow.keras.callbacks.EarlyStopping(monitor='val_loss',  patience = 10),
             tensorflow.keras.callbacks.ModelCheckpoint(filepath='plant_disease_aug1600_best_my_model_100.h5',
                                                        monitor='val_loss',
                                                        save_best_only=True)]

history = model.fit(augment.flow(x_train, y_train, batch_size = BATCH_SIZE),
                    validation_data=(x_test, y_test),
                    steps_per_epoch=len(x_train) // BATCH_SIZE,
                    epochs=EPOCHS,
                    verbose=1,
                    callbacks=callbacks)
```

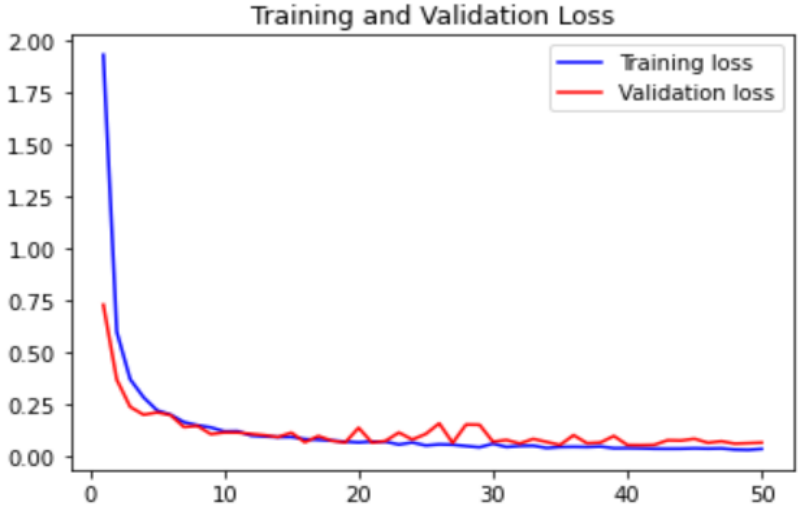
4. 모델설명

```
Epoch 36/100
665/665 [==============================] - 115s 173ms/step - loss: 0.0495 - accuracy: 0.9843 - precision: 0.9849 - recall: 0.9839
- val_loss: 0.0647 - val_accuracy: 0.9836 - val_precision: 0.9842 - val_recall: 0.9828
Epoch 37/100
665/665 [==============================] - 115s 173ms/step - loss: 0.0375 - accuracy: 0.9873 - precision: 0.9879 - recall: 0.9869
- val_loss: 0.1113 - val_accuracy: 0.9717 - val_precision: 0.9727 - val_recall: 0.9712
Epoch 38/100
665/665 [==============================] - 116s 174ms/step - loss: 0.0515 - accuracy: 0.9821 - precision: 0.9830 - recall: 0.9815
- val_loss: 0.0625 - val_accuracy: 0.9858 - val_precision: 0.9863 - val_recall: 0.9853
Epoch 39/100
665/665 [==============================] - 117s 176ms/step - loss: 0.0394 - accuracy: 0.9873 - precision: 0.9876 - recall: 0.9869
- val_loss: 0.0927 - val_accuracy: 0.9775 - val_precision: 0.9779 - val_recall: 0.9769
Epoch 40/100
665/665 [==============================] - 117s 175ms/step - loss: 0.0434 - accuracy: 0.9866 - precision: 0.9871 - recall: 0.9861
- val_loss: 0.0662 - val_accuracy: 0.9832 - val_precision: 0.9842 - val_recall: 0.9829
Epoch 41/100
665/665 [==============================] - 117s 176ms/step - loss: 0.0306 - accuracy: 0.9901 - precision: 0.9905 - recall: 0.9895
- val_loss: 0.0795 - val_accuracy: 0.9804 - val_precision: 0.9812 - val_recall: 0.9799
```

- Adagrad와 RMSprop의 장점을 결합한 옵티마이저 'Adam' 사용
- 다범주 데이터를 사용했기에 'categorical_crossentropy' 사용
- 'EarlyStopping'을 통해 불필요한 훈련을 제거
  ->41번째 훈련을 통해 모델훈련 마무리
- Accuracy, Precision, Recall이 각각 0.9901, 0.9905, 0.9895 등 훈련이 매우 잘 됐 다는 것을 파악

4. 모델설명_모델 정확도 평가



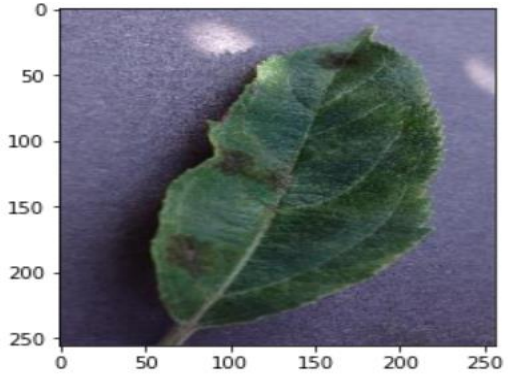


- 시간이 지날수록 정확도는 훈련 데이터, 테스트 데이터에서 차이가 거의 없었음
- 역시나 오차도 훈련데이터와 테스트 데이터의 그래프는 비슷한 흐름을 확인

4. 모델설명_모델 테스트(Plant Village)

```
def predict_disease(image_path):
    image_array = convert_image_to_array(image_path)
    np_image = np.array(image_array, dtype=np.float16) / 225.0
    np_image = np.expand_dims(np_image, 0)
    plt.imshow(plt.imread(image_path))
    result = keras_model_best.predict(np_image)
    result_top = np.argmax(keras_model_best.predict(np_image), axis=1)
    print("Top prob. : {0:.6f}%".format(result[0, result_top][0] * 100))
    print("Top prob._label : {}".format(label_binarizer.classes_[result_top][0]))
```

```
predict_disease(image_path_list[15])

Top prob. : 99.994385%
Top prob._label : Apple___Apple_scab
```



- 정확도가 99.994385%로 매우 높은 수치를 확인 가능

4. 모델설명_모델 성능 평가

- 모델평가 Acccuracy

```
print("[INFO] Calculating model accuracy")
keras_model_best = tensorflow.keras.models.load_model ₩
('plant_disease_aug1500_best_my_model_50.h5')
scores = keras_model_best.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

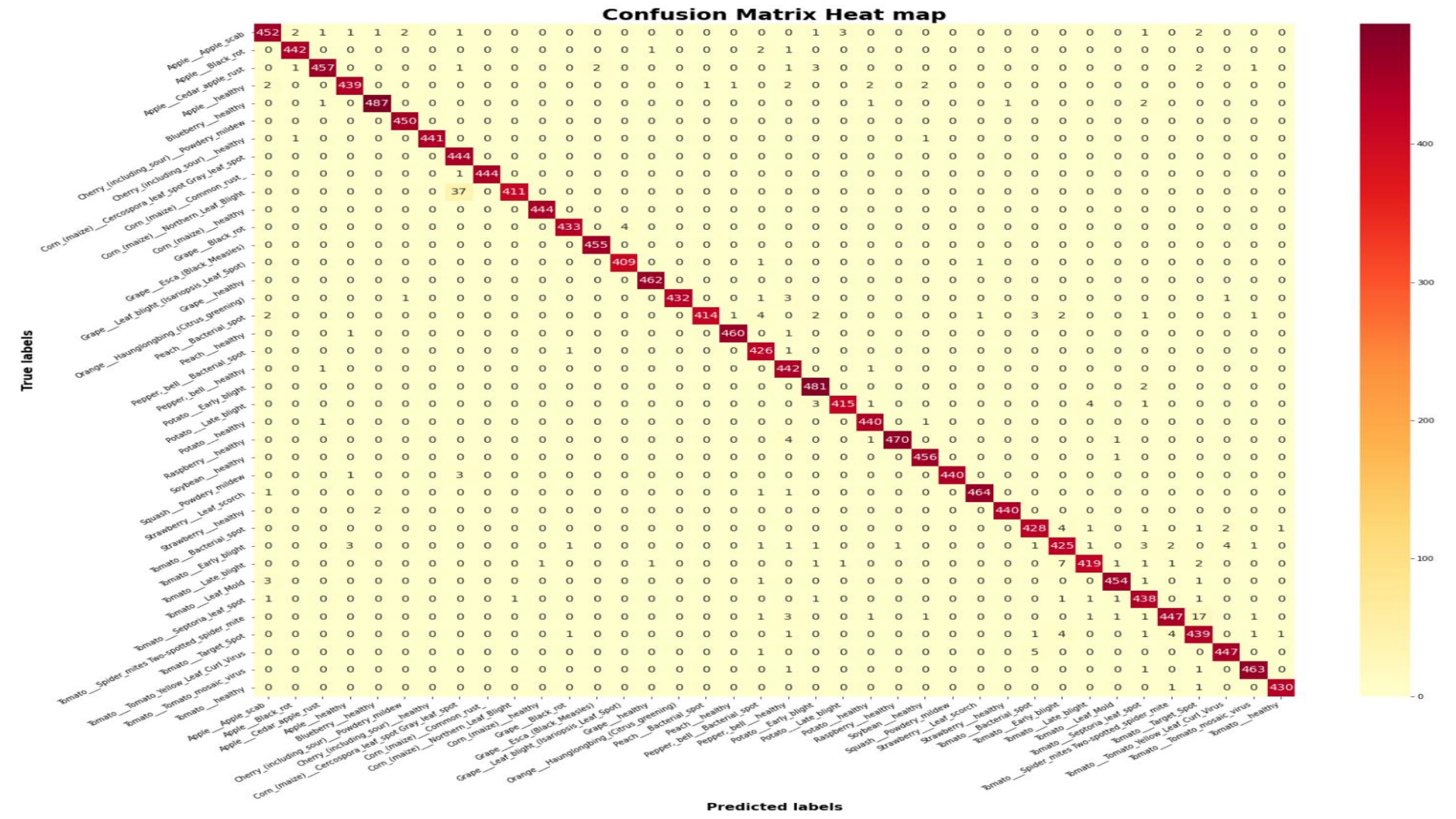
- 모델평가 Confusion Matrix (Precision,Recall,F1_score)

```
from sklearn.metrics import classification_report

# predict
pred = keras_model_best.predict(x_test, batch_size = BATCH_SIZE)
pred = np.argmax(pred, axis=1)
# label
y_target = np.argmax(y_test, axis=1)
y_target

print(classification_report(y_target, pred, target_names = list_diseases))
```

	precision	recall	f1-score	support
Apple___Apple_scab	0.98	0.97	0.97	467
Apple___Black_rot	0.99	0.99	0.99	446
Apple___Cedar_apple_rust	0.99	0.98	0.98	468
Apple___healthy	0.99	0.98	0.98	449
Blueberry___healthy	0.99	0.99	0.99	492
Cherry_(including_sour)___healthy	0.99	1.00	1.00	450
Cherry_(including_sour)___Powdery_mildew	1.00	1.00	1.00	443
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.91	1.00	0.95	444
Corn_(maize)___Common_rust_	1.00	1.00	1.00	445
Corn_(maize)___healthy	1.00	0.92	0.96	448
Corn_(maize)___Northern_Leaf_Blight	1.00	1.00	1.00	444
Grape___Black_rot	0.99	0.99	0.99	437
Grape___Esca_(Black_Measles)	1.00	1.00	1.00	455
Grape___healthy	0.99	1.00	0.99	411
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00	462
Orange___Haunglongbing_(Citrus_greening)	1.00	0.99	0.99	438
Peach___Bacterial_spot	1.00	0.96	0.98	431
Peach___healthy	1.00	1.00	1.00	462
Pepper,_bell___Bacterial_spot	0.97	1.00	0.98	428
Pepper,_bell___healthy	0.96	1.00	0.98	444
Potato___Early_blight	0.98	1.00	0.99	483
Potato___healthy	0.99	0.98	0.98	424
Potato___Late_blight	0.98	1.00	0.99	442
Raspberry___healthy	1.00	0.99	0.99	476
Soybean___healthy	0.99	1.00	0.99	457
Squash___Powdery_mildew	1.00	0.99	1.00	444
Strawberry___healthy	1.00	0.99	0.99	467
Strawberry___Leaf_scorch	1.00	1.00	1.00	442
Tomato___Bacterial_spot	0.98	0.98	0.98	438
Tomato___Early_blight	0.96	0.96	0.96	445
Tomato___healthy	0.98	0.96	0.97	435
Tomato___Late_blight	0.99	0.99	0.99	460
Tomato___Leaf_Mold	0.96	0.98	0.97	445
Tomato___Septoria_leaf_spot	0.98	0.94	0.96	474
Tomato___Spider_mites Two-spotted_spider_mite	0.94	0.97	0.95	453
Tomato___Target_Spot	0.98	0.99	0.99	453
Tomato___Tomato_mosaic_virus	0.99	0.99	0.99	466
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	1.00	1.00	432
accuracy			0.98	17100
macro avg	0.99	0.98	0.98	17100
weighted avg	0.99	0.98	0.98	17100

- 데이터가중을 2번해서 각각의 식물들의 정확도 재현율, F1-Score 등이 매우 높은 수치를 기록
- 제일 낮은 수치가 0.91일 정도로 전체적으로 0.9이상의 수치 확인

Accuracy: 0.9854714912280702
Precision: 0.9857894730621745
Recall: 0.9854856366579019
F1_score: 0.9855091124181911


Confusion Matrix Heat map
True labels
Predicted labels
Apple___Apple_scab
Apple___Black_rot
Apple___Cedar_apple_rust
Apple___healthy
Blueberry___healthy
Cherry_(including_sour)___Powdery_mildew
Cherry_(including_sour)___healthy
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot
Corn_(maize)___Common_rust_
Corn_(maize)___Northern_Leaf_Blight
Corn_(maize)___healthy
Grape___Black_rot
Grape___Esca_(Black_Measles)
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Grape___healthy
Orange___Haunglongbing_(Citrus_greening)
Peach___Bacterial_spot
Peach___healthy
Pepper,_bell___Bacterial_spot
Pepper,_bell___healthy
Potato___Early_blight
Potato___Late_blight
Potato___healthy
Raspberry___healthy
Soybean___healthy
Squash___Powdery_mildew
Strawberry___Leaf_scorch
Strawberry___healthy
Tomato___Bacterial_spot
Tomato___Early_blight
Tomato___Late_blight
Tomato___Leaf_Mold
Tomato___Septoria_leaf_spot
Tomato___Spider_mites Two-spotted_spider_mite
Tomato___Target_Spot
Tomato___Tomato_Yellow_Leaf_Curl_Virus
Tomato___Tomato_mosaic_virus
Tomato___healthy
400
300
200
100
0

1. Corn_(maize)___
Northern_leaf_Blight /
Corn_(maize)___
Cercospora_leaf_spot Gray_leaf_spot

	Northen_Leaf	Cercospora_leaf
1		
2		

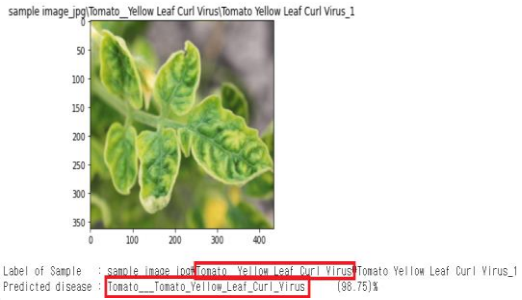
2. Tomato___Target_Spot /
Tomato___
Spider_mites Two-spotted_spider_mite

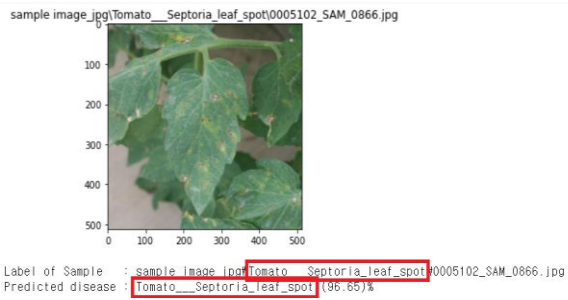
	Target_Spot	Spider_mites
1		
2		

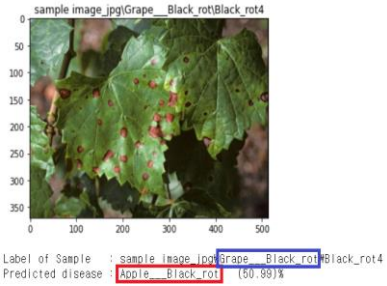
- 처음 이미지_size를 256*256에서 128*128로 줄인 것 때문에 특징을 제대로 잡지 못해 해당 범주들을 잘 분류하지 못했음

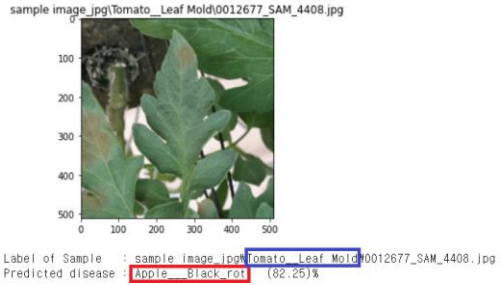
4. 모델설명_모델 테스트(실제 작물이미지)

```
### 분류(Test)

while True:
    pred = input('Input 1[predict disease] or any key[Quit] : ')
    if pred == '1':
        sample_image_label = random.choice(glob('./PlantVillage/val/*'))
        sample_image_name = random.choice(glob(sample_image_label + '/*.JPG'))
        predict_disease(sample_image_name)
    else:
        break
```









- 10개의 실제 사진 예측을 해본 결과 2개의 사진은 동일 항목에 대해서 예측했지만 나머지 8개의 실제 사진은 이에 해당하는 품목들을 예측을 하지 못해 Overfitting이 발생했음

5. 분석결과 고찰

- 처음 목적을 정한 후, 이에 맞는 데이터 셋을 구성해야 된다는 것을 알 수 있음

- 이는 학습데이터셋의 이미지가 인물사진으로 비유하면 증명사진에 해당되는 이미지이기 때문에 다수의 잎이 있는 경우나 배경이 있는 경우에는 학습되지 않아 오류가 발생함을 알 수 있었고 이는 차후 데이터셋 구성함에 있어 꼭 반영해야 하는 요소임을 깨닫게 됨

- 데이터 모델을 훈련시킬 때, 각 범주마다 비슷한 개수의 데이터를 갖고 있으면 성능이 좋다는 것을 알 수 있음

6. 작물진단시스템 활용

- 작물(토마토) 질병 진단 시스템은 웹기반 UI로 고도화를 통하여 농업현장에 실용적 활용이 가능하다.
- 추후 병충해진단 A.I드론, 곤충로봇을 구축하여 진단시스템을 적용을 시켜서 나중에 병충해에 대한 전문지식 및 인력 부족과 같은 문제에 해결책 방안 도출
- 동일한 인공지능 기법을 적용하여 대상작목의 확대가 가능하다.

7. 참고자료

- 딥러닝 기반 토마토 병충해 분류 시스템 연구
- 심층 CNN 기반 구조를 이용한 토마토 작물 병해충 분류 모델
- 딥러닝 학습 향상을 위한 고려 사항들 – GIS Developer
- Adam — 딥 러닝 최적화의 최신 트렌드. (ichi.pro)
- 사이언스모니터 | The Science Monitor - 농수산업에도 AI 로봇, 병충해 감시-작물수확
- YTN 사이언스' 보기 - 장애물에 부딪혀도 추락하지 않는 비행로봇의 비밀
- 성공적인 인공지능·머신러닝 모델을 위한 데이터 관리