Cassava Leaf Disease Classification

수학과 오서영

Overview

Introduction

Cassava is a key food security crop in Africa
But **viral diseases** are major sources of poor yields.
Existing methods of disease detection require experts to visually inspect and diagnose the plants -> labor-intensive, low-supply and costly.

So I want to solve this problem through **data science**.
-> classify each cassava image into **5** category

1. Data Exploration and Visualization

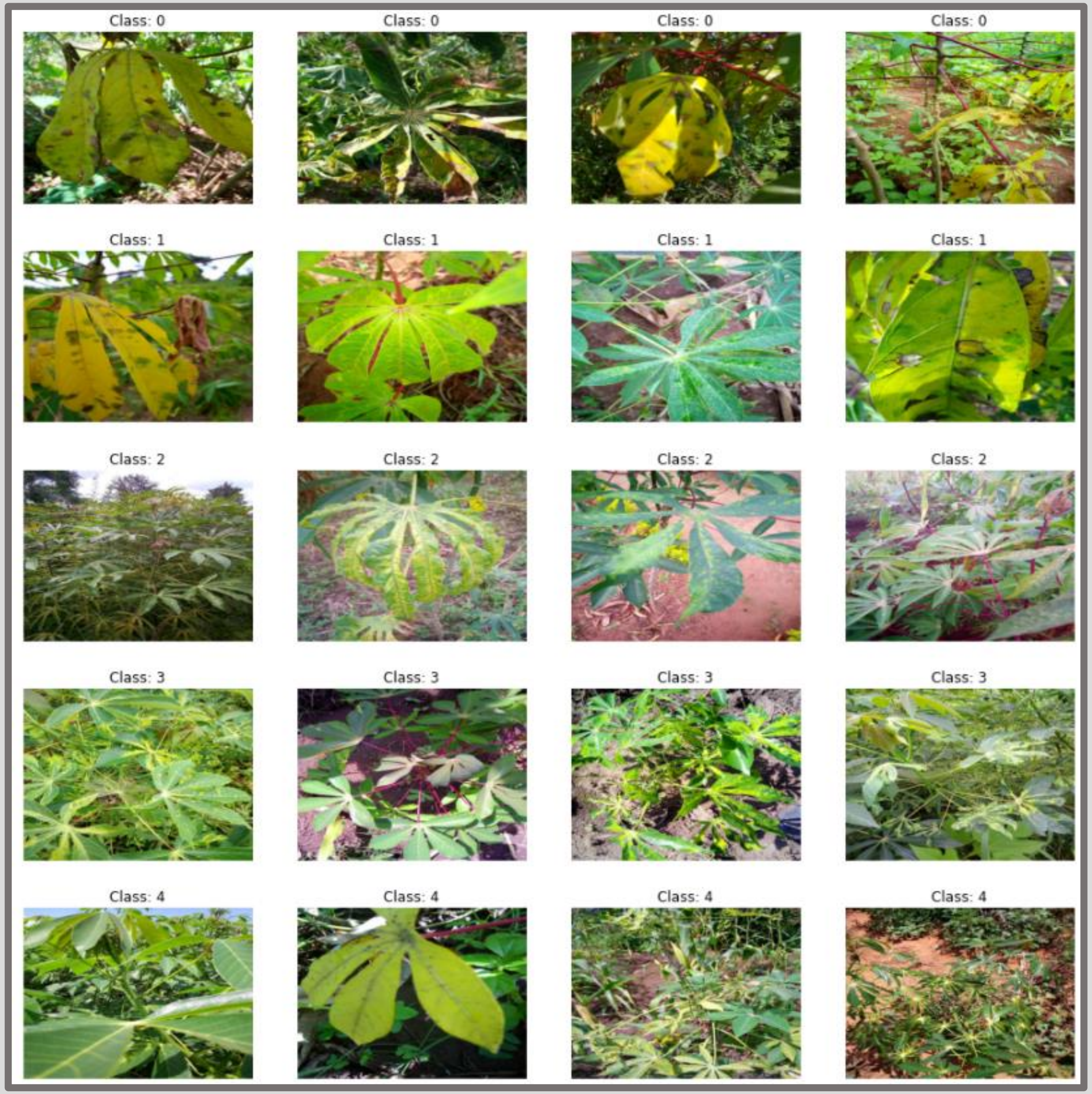
```python
BASE_DIR = os.getcwd()
with open(os.path.join(BASE_DIR, "label_num_to_disease_map.json")) as file:
    map_classes = json.loads(file.read())
    map_classes = {int(k) : v for k, v in map_classes.items()}

print(json.dumps(map_classes, indent=4))
```

```
{
    "0": "Cassava Bacterial Blight (CBB)",
    "1": "Cassava Brown Streak Disease (CBSD)",
    "2": "Cassava Green Mottle (CGM)",
    "3": "Cassava Mosaic Disease (CMD)",
    "4": "Healthy"
}
```

```python
img_shapes = {}
for image_name in os.listdir(os.path.join(BASE_DIR, "train_images"))[:300]:
    image = cv2.imread(os.path.join(BASE_DIR, "train_images", image_name))
    img_shapes[image.shape] = img_shapes.get(image.shape, 0) + 1

print(img_shapes)
```

```
{(600, 800, 3): 300}
```

1. Data Exploration and Visualization

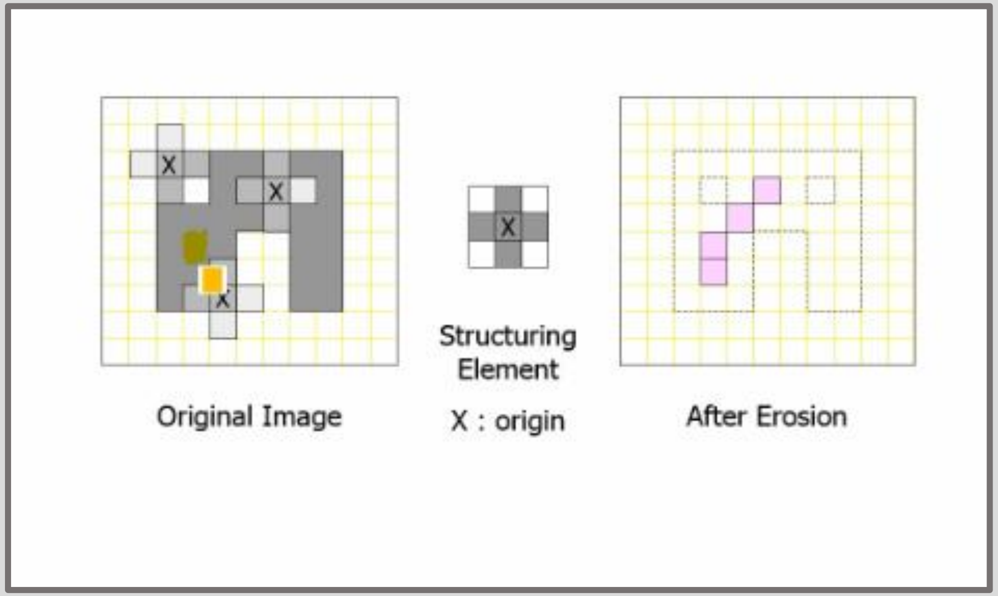


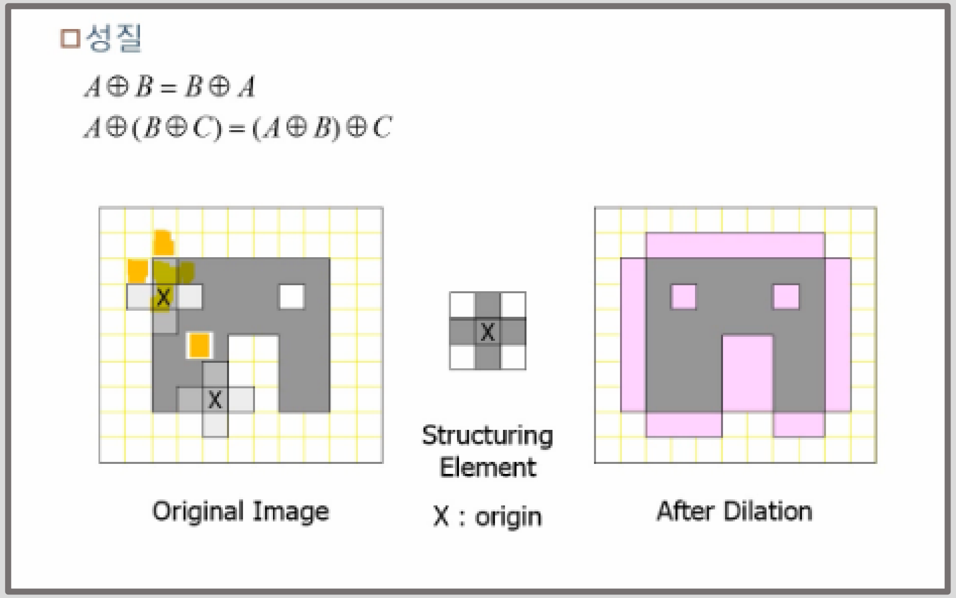

```
input_files = os.listdir(os.path.join(BASE_DIR, "train_images"))
print(f"Number of train images: {len(input_files)}")
```

```
Number of train images: 21397
```

2. Data Preprocessing

Segmentation with Opencv





Erosion

: 각 Pixel에 structuring element를 적용하여 안겹치는 부분이 하나라도 있으면 그 중심 pixel을 제거하는 방법

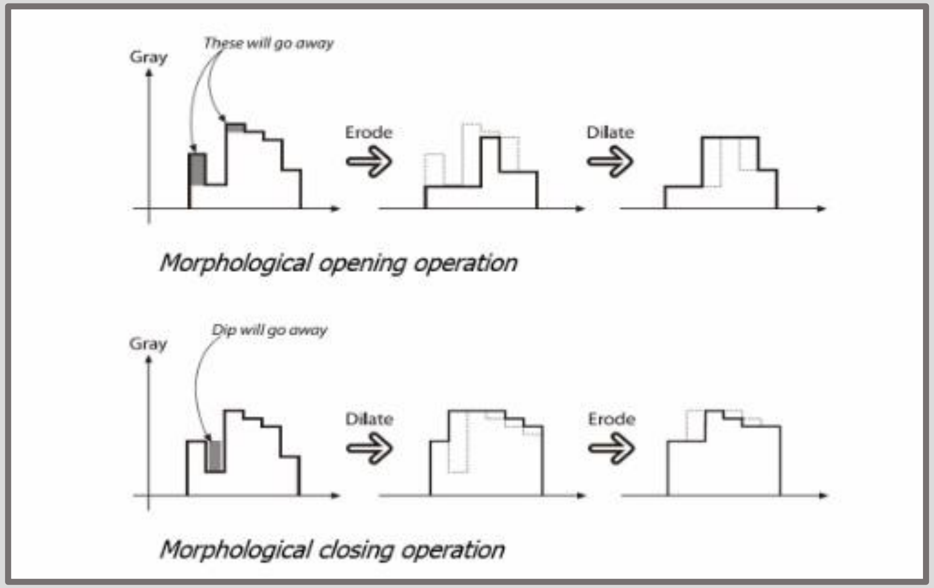
Dilation

: 각 pixel에 structuring element를 적용하여 겹치는 부분이 하나라도 있으면 이미지를 확장

2. Data Preprocessing

Segmentation with Opencv

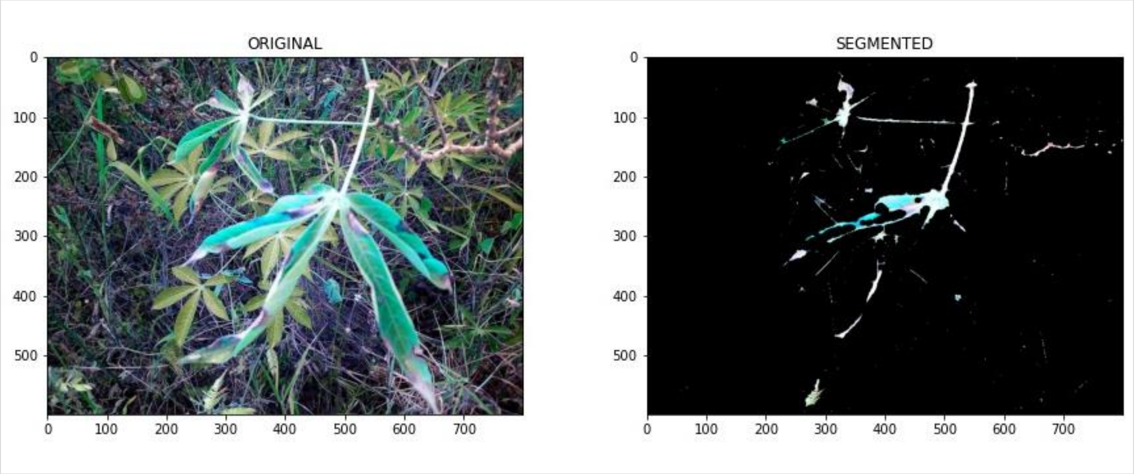
```python
def read_image(image_id , label):
    plt.figure(figsize=(15, 10))
    image = cv2.imread(os.path.join(BASE_DIR, "train_images", image_id))

    return image

def create_masks(image):
    image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_hsv = np.array([0,0,250])
    upper_hsv = np.array([250,255,255])

    mask = cv2.inRange(image_hsv, lower_hsv, upper_hsv)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11,11))
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
    return mask

def segment_image(image):
    mask = create_masks(image)
    output = cv2.bitwise_and(image, image, mask=mask)
    return output/255
```

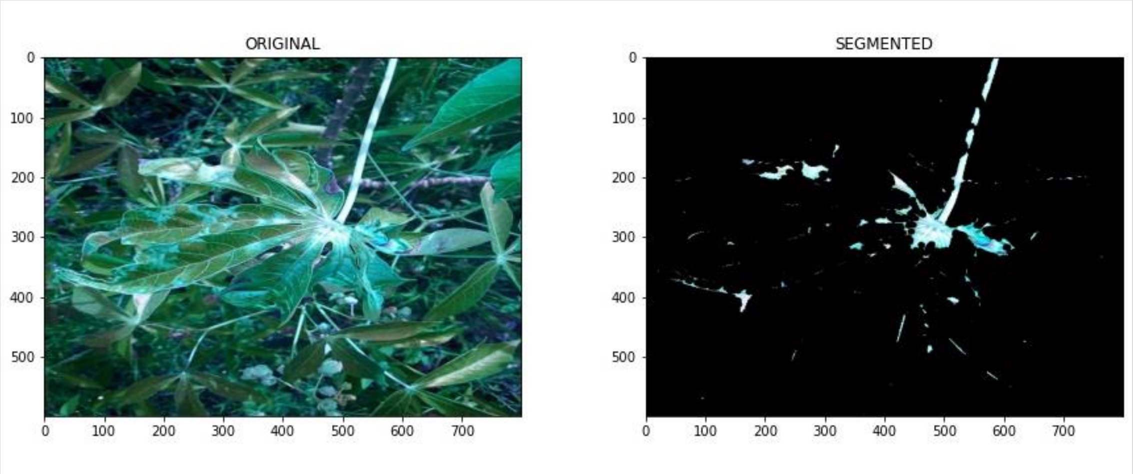


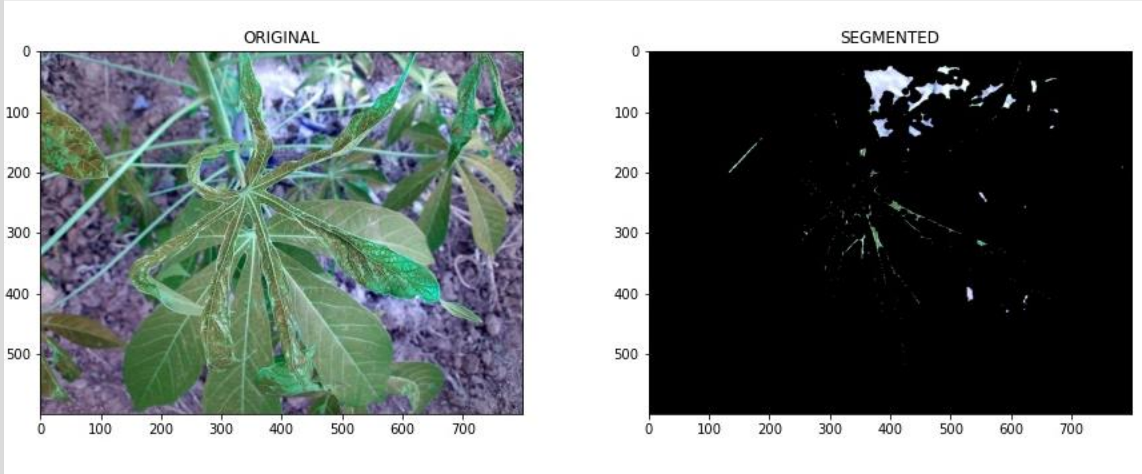
Opening & Closing

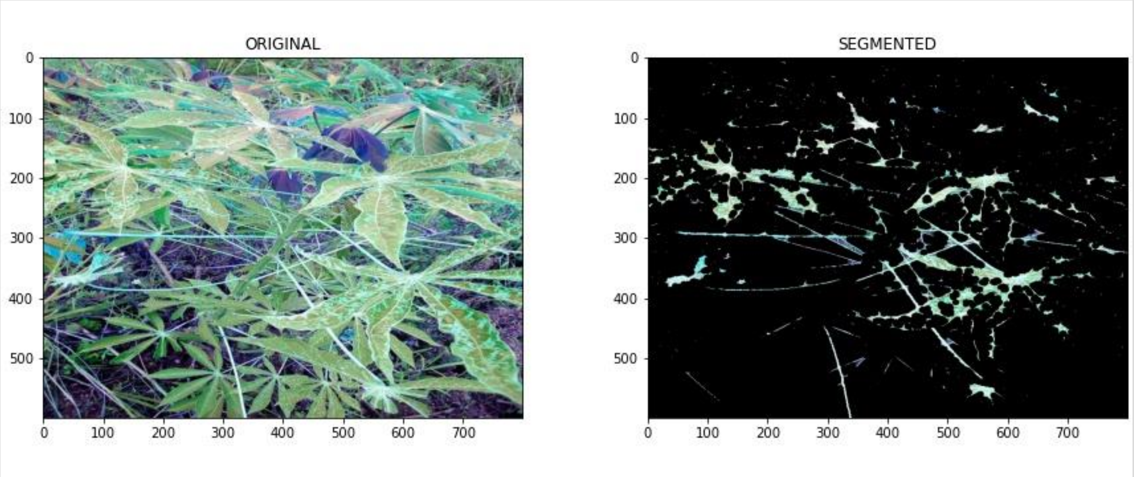
2. Data Preprocessing

Segmentation -> X









2. Data Preprocessing

Image Augmentation + GrayScale

```python
train_labels.label = train_labels.label.astype('str')

train_datagen = ImageDataGenerator(validation_split = 0.2,
                                   rescale = 1./255,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   shear_range = 0.1)

train_generator = train_datagen.flow_from_dataframe(train_labels,
                         directory = os.path.join(BASE_DIR, "train_images"),
                         subset = "training",
                         x_col = "image_id",
                         y_col = "label",
                         target_size = (target_size, target_size),
                         batch_size = batch_size,
                         color_mode = 'grayscale',
                         class_mode = "categorical")
```

3. Training – Vanilla CNN

```
model = Sequential()
model.add(Conv2D(64, kernel_size=(3, 3), strides=(1, 1), padding='same',
                 activation='relu',
                 input_shape=(32, 32, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 32, 32, 64)        1792
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 32)        18464
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               524544
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 1285
=================================================================
Total params: 546,085
Trainable params: 546,085
Non-trainable params: 0
_________________________________________________________________
```

Test accuracy
less than 60%

3. Training - EfficientNet

The traditional way to improve CNN performance
: increase the depth of model

-> **EfficientNet**
: consider not only **depth**
but also **width** and **resolution** together

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Information : 128 x 128 > 64 x 64.
But high-resolution input can result in a large calculation cost,
and does not provide a significant benefit to model performance after certain values.
-> **Balancing** the width, depth, and resolution of a network is very important for improving performance.

Quantitative Analysis

Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International Conference on Machine Learning*. PMLR, 2019.

3. Training - EfficientNet

```
early_stopping = EarlyStopping(monitor='val_accuracy', patience=5)
mc = ModelCheckpoint('best_model.h5', monitor='val_loss', mode='min', save_best_only=True)
```

```
model = Sequential()
optimizer = Adam(lr=0.00105)

b4model = EfficientNetB4(include_top = False,
                         weights = None,
                         pooling='avg')
model.add(b4model)
model.add(Dense(5, activation ='softmax'))

model.summary()
```

```
model.add(Dense(256, activation = 'relu'))
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnetb4 (Functional)  (None, 1792)              17673823
_________________________________________________________________
dense_11 (Dense)             (None, 5)                 8965
=================================================================
Total params: 17,682,788
Trainable params: 17,557,581
Non-trainable params: 125,207
_________________________________________________________________
```

3. Training - EfficientNet

```
model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
```

```
%%time
hist10 = model.fit(
    train_generator,
    steps_per_epoch = steps_per_epoch,
    epochs = epochs,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    callbacks=[early_stopping, mc],
    verbose = 1)
```

Test accuracy
79~81%

Training on a single epoch is extremely slow… (3 hours)
-> GPU?

Kaggle Competition

[1] Cassava Leaf Disease Classification,
https://www.kaggle.com/c/cassava-leaf-disease-classification/overview

References

[1] The shortest way to Tensorflow baseline,
https://www.kaggle.com/nozarchos/the-shortest-way-to-tensorflow-baseline

[2] Tensorflow ViT and Image Pre-processing,
https://www.kaggle.com/digvijayyadav/tensorflow-vit-and-image-pre-processing

[3] Convolutional Neural Network의 성능을 높이는 현명한 방법 : EfficientNet Google AI,
https://ichi.pro/ko/convolutional-neural-networkui-seongneung-eul-nop-ineun-hyeonmyeonghan-bangbeob-efficientnet-google-ai-220426838454404