

5 Flower Types Classification

Keras를 활용한 인공지능망 만들기

Kim.TH

목 차

5 Flower Types Classification

1. Project 개요 및 Dataset 소개

2. 분류(Classification)?

2.1 분류(Classification)란?

2.2 인공지능망에서 Image data 처리 과정

3. Project 진행 계획

4. 라이브러리 소개 및 활용

4.1 Keras

4.2 openCV

4.3 Numpy

4.4 Matplotlib

5. Project 진행 과정

5.1 합성곱 층 필터 수 결정

5.2 밀집 층 노드 수 결정

5.3 합성곱 층 수 결정

5.4 밀집 층 수 결정

5.5 Hyperparameter 조정

5.6 최종 신경망

6. 결과 및 고찰

1. Project 개요 및 Dataset 소개

Project 개요 및 Dataset 소개

Project 개요

- Project 목적 : Keras를 활용한 이미지 분류
- Project 목표 : 90% 이상 정확도를 가진 신경망 설계

Dataset : 5 Flower Types Classification

- 출처 : kaggle
- 5종류의 꽃 사진(폴더 별로 구분)
 - 1 - Lilly : 은방울꽃 : 1000개 파일
 - 2 - Lotus : 연꽃 : 1000개 파일
 - 3 - Sunflower : 해바라기 : 1000개 파일
 - 4 - Orchid : 난초 : 1000개 파일
 - 5 - Tulip : 튤립 : 1000개 파일
- train, validation, test 구별 없고, image 크기도 다른 총 5000장의 이미지 파일



Lilly



Lotus

Sunflower



Orchid

Tulip

2. 분류(Classification)?

2.1. 분류(Classification)란?

2.2. 인공신경망에서 이미지 data 처리 과정

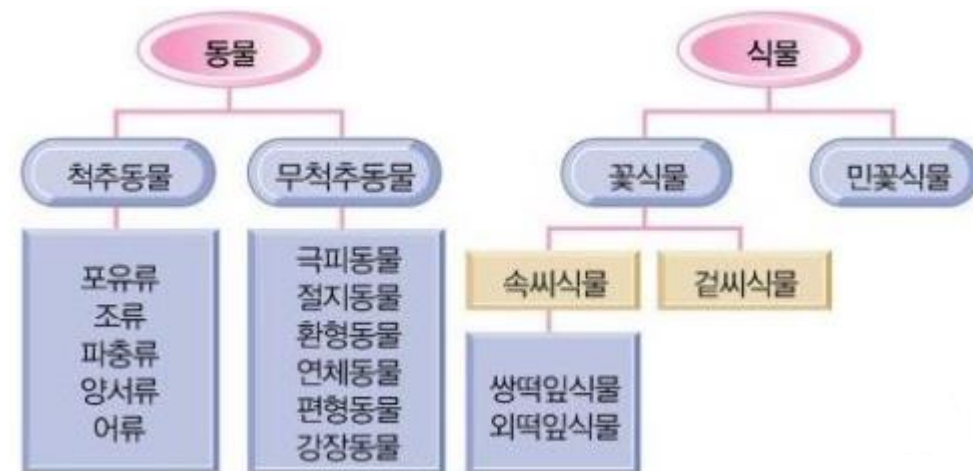
2.1. 분류(Classification)란?

분류(Classification)란?

- 종류에 따라서 가름
- 특정 데이터를 어떠한 기준에 따라 여러 범주(클래스)로 구분
 - 이진 분류(Binary Classification)
: 주어진 데이터를 두 범주로 구분(Yes or No)
 - 다중 분류(Multi-class Classification)
: 주어진 데이터를 셋 이상의 범주로 구분



[Pict2.1. 이진 분류]

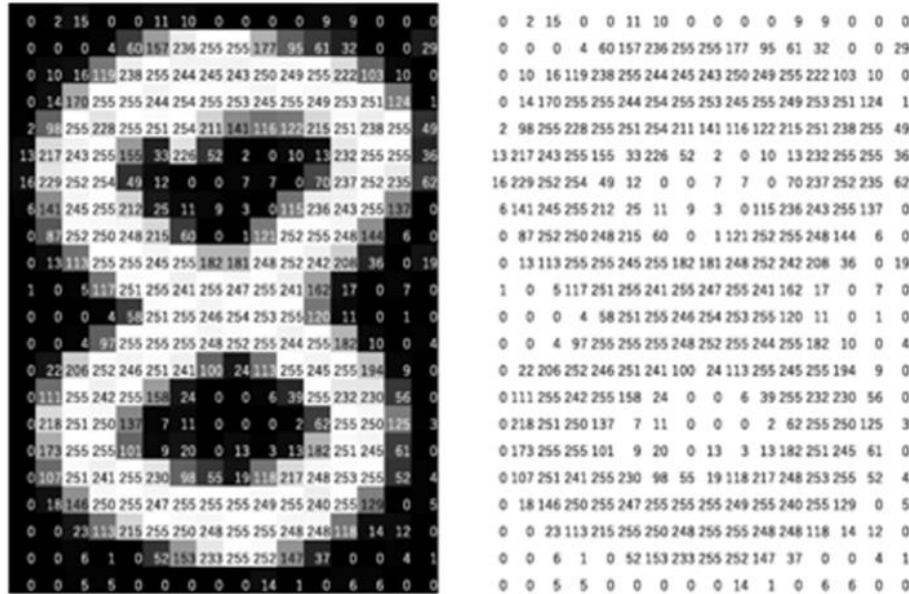


[Pict2.2. 다중 분류]

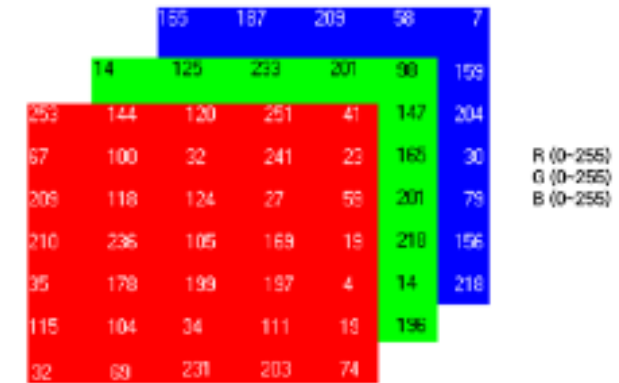
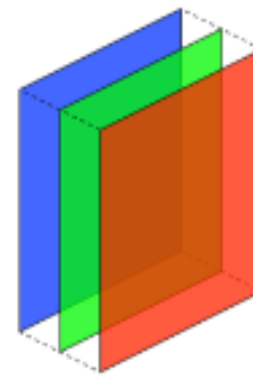
2.2. 인공신경망에서 이미지 data 처리 과정 - Image data

Image Data

- Image는 pixel로 이루어짐
- 하나의 pixel은 0(검정색)~255(흰색) 범위의 정수 값을 가짐
- color Image일 경우 RGB(빨강, 초록, 파랑)에 대해 은 0~255 범위의 정수 값을 가짐
- pixel 값 = Image Data = 신경망의 입력값



[Pict2.3. 흑백 Image Data]

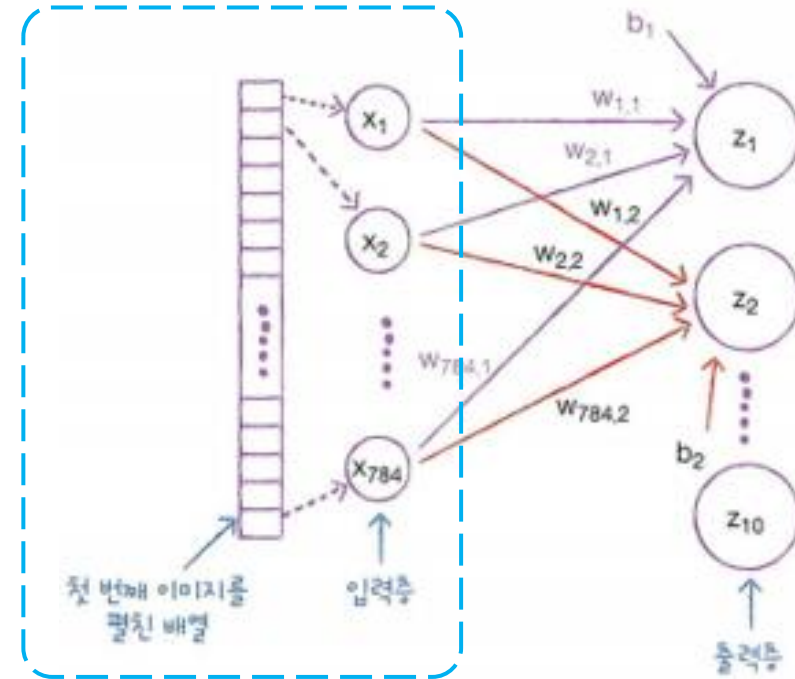


[Pict2.4. 칼라 Image Data]

2.2. 인공신경망에서 이미지 data 처리 과정 - Input data

Image input data

- 컴퓨터는 2차원, 3차원을 인식하지 못함
=> 1차원으로 변환
=> 출력값(z) = $\sum_{i=1}^n (\text{가중치}(W_i) \times \text{입력값}(\text{pixel})) + \text{편향}(b)$
=> 출력값 = 다음 층의 입력값
=> 마지막 출력층의 출력값 중 가장 높은 값을 정답으로 예측
- 입력값 많을수록 연산 속도 느려짐
=> 같은 pixel의 흑백 image에서 컬러 image로 연산 시 data 수는 3배가 됨
=> 100x100 Image : 흑백(1만 개 data), 컬러(3만 개 data)
=> 1000x1000 Image : 흑백(100만 개 data), 컬러(300만 개 data)
- 학습 data 많으므로 신경망 학습 중 과대적합 가능성 높아짐
- 대안 => CNN 알고리즘

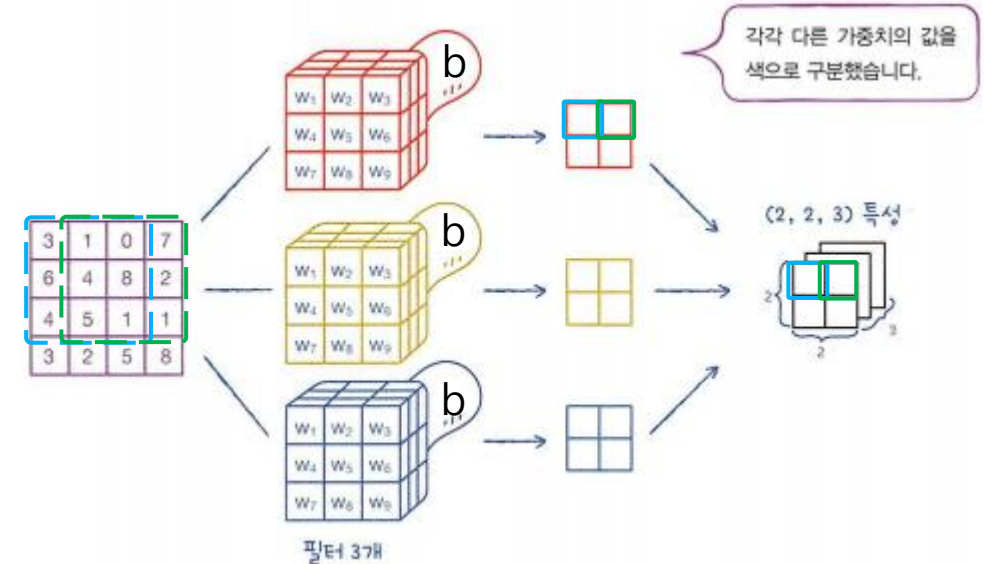


[Pict2.5. Image Data]

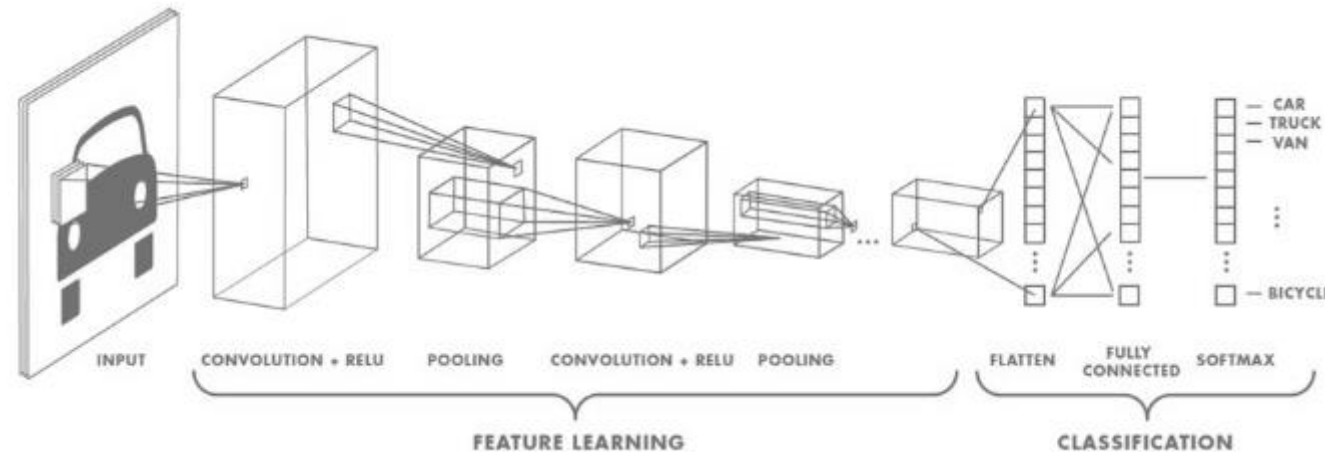
2.2. 인공신경망에서 이미지 data 처리 과정 - CNN

CNN(Convolutional Neural Network) : 합성곱 신경망

- 커널(kernel)을 이용하여 2차원의 특성맵을 생성하며 학습
- 커널(kernel), 풀링(pooling), 활성화함수 등을 이용하여 이미지 특징은 최대한 유지하면서 입력값을 줄일 수 있음
- 이미지 특징 파악 시 이미지의 차원 변환 없이 그대로 계산되므로 이미지의 특징 파악이 더 잘 됨
=> 더 나은 가중치와 편향 값 찾을 가능성 ↑
- 따라서 이미지 data 학습에 CNN 알고리즘이 주로 쓰임



[Pict2.6. 합성곱으로 특성맵 생성]



[Pict2.7. Image 처리 과정(CNN)]

3. Project 진행 계획

3. Project 진행 계획

무엇을 정해야 하는가?

<조작 변인 - 은닉층>

- 노드 수(합성곱 층 필터 수, 밀집 층 노드 수)
- 층 수(합성곱 층 수, 밀집 층 수)

<통제 변인>

- Image 크기 : 160 x 160
- Hyperparameter 값
 - 합성곱 층마다 적용
 - kernel_size = 3x3
 - padding = 'same'
 - pooling = MaxPooling2D_(2x2)
 - 밀집 층마다 적용
 - dropout = 0.2
 - activation = 은닉층 : 'relu'
출력층 : 'softmax'
 - 신경망 설정 값
 - optimizer = 'adam'
 - loss='sparse_categorical_crossentropy'
 - batch size = 32(기본값)
 - epochs = 15

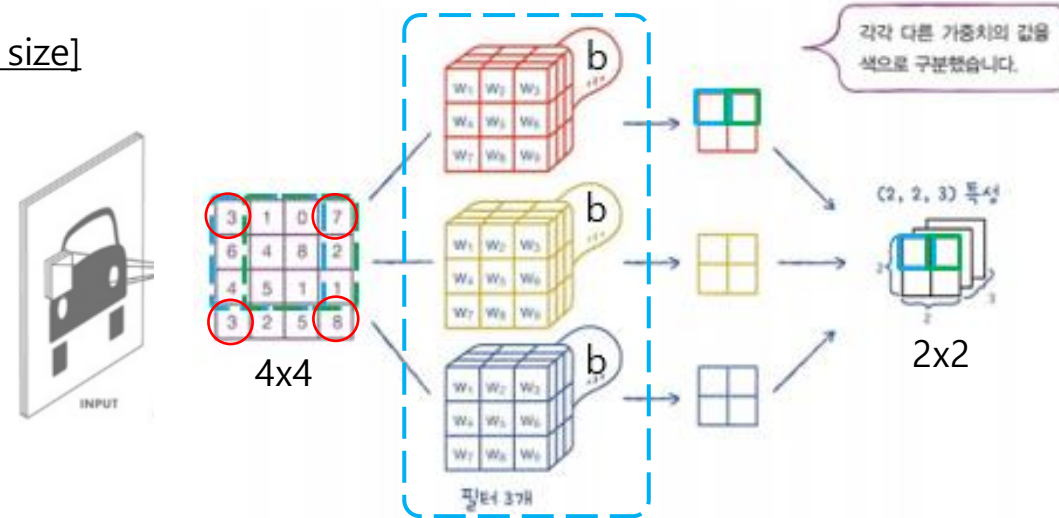
과정	조작 변인	조작 내용	통제 변인
1	합성곱 층 필터 수	32 / 64 / 128 / 256	· 좌측 통제 변인 값 · 합성곱 층 수 = 1 · 밀집 층 수 = 1 · 밀집 층 노드 수 = 50
2	밀집 층 노드 수	25 / 50 / 100 / 200	· 좌측 통제 변인 값 · 합성곱 필터 수 = 1과정 결정 · 합성곱 층 수 = 1 · 밀집 층 수 = 1
3	합성곱 층 수	1 / 3 / 5	· 좌측 통제 변인 값 · 합성곱 필터 수 = 1과정 결정 · 밀집 노드 수 = 2과정 결정 · 밀집 층 수 = 1
4	밀집층 층 수	1 / 2 / 3	· 좌측 통제 변인 값 · 합성곱 필터 수 = 1과정 결정 · 밀집 노드 수 = 2과정 결정 · 합성곱 층 수 = 3과정 결정
5	은닉층의 필터, 노드 수와 층 수들이 정해져 대략적인 신경망 구조의 학습 결과를 본 후 Hyperparameter 값 조정(detail 조정)		

3. Project 진행 계획 - Hyperparameter 값

통제 변인 Hyperparameter 값

- kernel_size = 3x3
- padding = 'same'
- pooling = MaxPooling2D_(2x2)

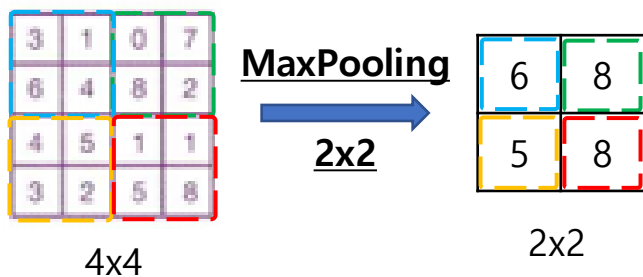
[Pict3.1. kernel size]



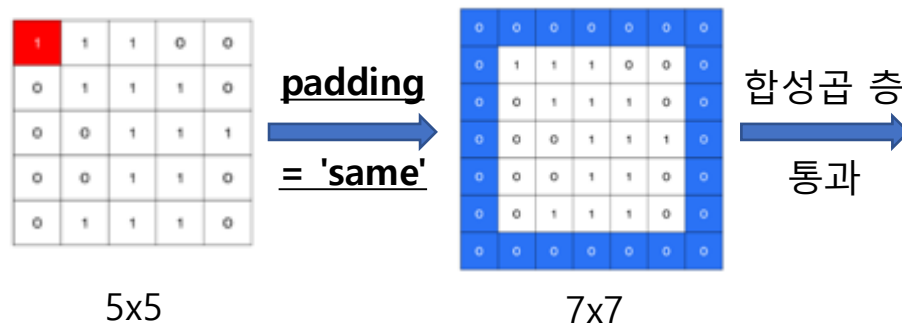
kernel size = 3x3

특성 맵 size = 입력 size - (커널 size - 1) ---- ex) 4 - (3 - 1) = 2

[Pict3.3. MaxPooling]



[Pict3.2. padding]



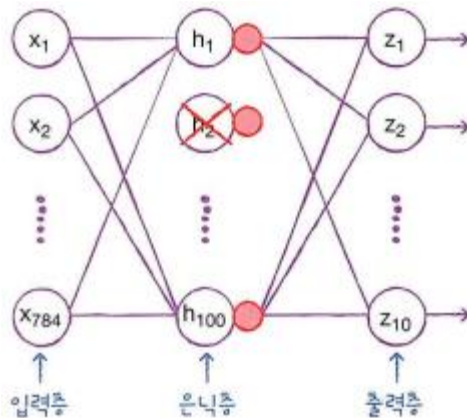
가로 : 7 - (3 - 1) = 5
세로 : 7 - (3 - 1) = 5
입력 size = 특성 맵 size
만약 kernel size가 5라면
padding 2번 해야 입출력 size 같아짐

3. Project 진행 계획 - Hyperparameter 값

통제 변인 Hyperparameter 값

- 밀집 층 dropout = 0.2

: 학습 과정에서 dropout이 적용된 층의
20%(랜덤)의 노드 출력을 0으로 만들
: 과대적합 해소에 도움



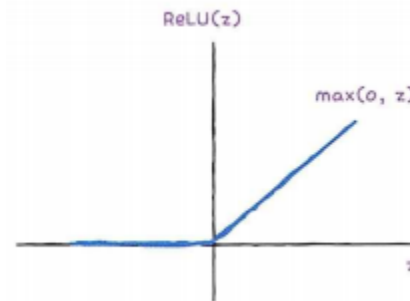
[Pict3.4. dropout]

통제 변인 Hyperparameter 값

- activation

- 은닉층(합성곱 층, 밀집 층) : 'relu' 함수

: 입력이 양수라면 그 값 그대로 통과 입력이 음수라면 그 값을 0으로 처리
: relu 함수는 이미지 처리에 특히 좋은 성능을 낸다고 알려짐



[Pict3.5. activation]

30	-10	20	40
20	30	-100	100
50	25	15	-15
80	-30	40	10

relu

30	0	20	40
20	30	0	100
50	25	15	0
80	0	40	10

- 출력층 : 'softmax' 함수

: 각각의 출력층 노드 값을 0~1사이로 압축하여
각 노드 전체 합이 1이 되도록 만들어 주는 함수
: 출력값 중 가장 높은 값을 정답으로 예측

$$S(P_i) = \frac{P_i}{\sum_{k=1}^K P_k}$$

$$C_k = \operatorname{argmax}_{1 \leq k \leq K} S(P_k)$$

3. Project 진행 계획 - Hyperparameter 값

통제 변인 Hyperparameter 값

● 신경망 설정 값

- optimizer = 'adam'

: 경사하강법을 기반으로 변형

: adam(ADaptive Moment estimation)

: Momentum(기울기 방향 보정)과

RMSprop(기울기 크기 조정)를 조합한 최적화 방법

: 다양한 분야의 데이터에 대하여

평균적으로 높은 성능을 보임

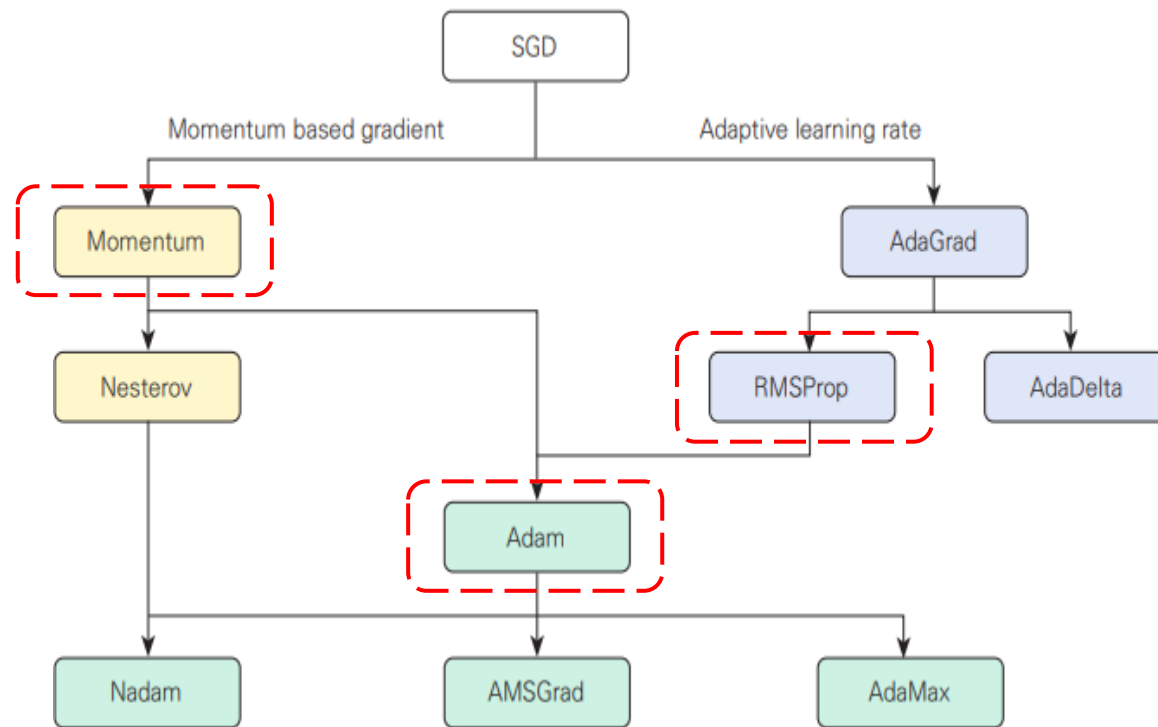
- batch_size = 32(기본값)

: 훈련 data 32개마다 가중치 조정

: keras는 batch_size 따로 기입하지 않으면 기본적으로

미니배치 경사하강법을 적용

batch_size=32로 훈련



[Pict3.6. optimizer]

4. 라이브러리 소개 및 활용

4.1. OpenCV

4.2. NumPy

4.3. matplotlib

4.4. Keras

4.1. 라이브러리 소개 및 활용 - OpenCV



- 컴퓨터 비전에 맞게 개발된
오픈 소스 라이브러리
- 영상 파일 입출력, 객체 탐지 등
영상 관련 기능 제공
- data 전처리
 - : 이미지 변수 지정, 크기 고정
 - : cv2.imread()로 이미지들을
160x160 크기로 변환하여
하나의 변수 'imgs'에 지정
 - : 이미지 data는 'list' type으로
[BGR] 순으로 저장되어 있음

```
import os
import cv2
def tran_img(path):
    imgs = []

    # os.listdir(path) : path 경로 안에 파일 명들을 list 요소로 반환.
    for fn in os.listdir(path):
        # cv2.imread('경로포함 파일명') : 주의) cv2로 이미지를 읽으면 RGB가 아닌 BGR로 읽어온다.
        img = cv2.imread(os.path.join(path, fn))
        if img is not None:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, (160, 160)) # 이미지 크기 변환
            imgs.append(img)
    return imgs
```

```
print(type(lilly_images))
print(lilly_images)
```



```
<class 'list'>
[array([[59, 52, 44],
        [59, 52, 44],
        [60, 53, 45],
        ...,
        [66, 60, 52],
        [60, 55, 51],
        [55, 51, 48]],
```


NumPy

- 파이썬 라이브러리
 - 행렬이나 다차원 배열 같은 수치 계산을 쉽게 할 수 있게 지원
 - TensorFlow는 NumPy 배열을 사용해서 신경망 층으로 신호 전달
-
- data 전처리
 - : np.array('data')
 - 넘파이 배열로 변환

```
import numpy as np
# np 배열 만들기
train_input = np.array(lilly_train_input + lotus_train_input + orchid_train_input + sunflower_train_input + tulip_train_input)
train_label = np.array(lilly_train_label + lotus_train_label + orchid_train_label + sunflower_train_label + tulip_train_label)
test_input = np.array(lilly_test_input + lotus_test_input + orchid_test_input + sunflower_test_input + tulip_test_input)
test_label = np.array(lilly_test_label + lotus_test_label + orchid_test_label + sunflower_test_label + tulip_test_label)
```

4.3. 라이브러리 소개 및 활용 - matplotlib



- 데이터 시각화에 활용되는
오픈 소스 라이브러리

- 딥러닝 활용 결과나
데이터 분석 결과를
그래프 등으로 시각화

- data 전처리

: plt.figure('size') - 시각화 영역 전체 size 지정

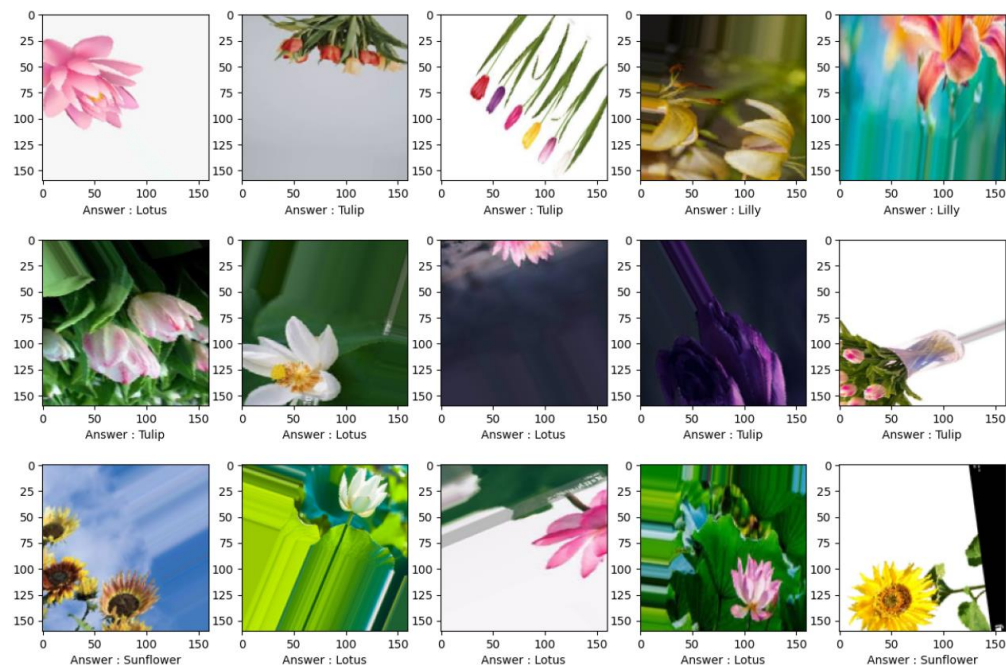
: plt.subplot(행, 열, 위치)

- 전체 size 중 위치 지정

: plt.imshow('data') - subplot으로 지정한
위치에 data 넣기

: plt.show() - 시각화 표시

```
import matplotlib.pyplot as plt
# train data idx 번째 이미지를 시각화
# 시각 그래프 총 크기 지정
plt.figure(figsize=(15,10))
for i in range(15):
    # 전체 크기에서 3행 5열로 칸 나눔
    plt.subplot(3,5,i+1)
    # 각각 이미지 1행 1열부터 표현.
    plt.imshow(images[i])
    # x축 아래 label 표시
    plt.xlabel(f"Answer : {labels_dict[np.argmax(labels[i])]}")
    # plt.title(f"Label: {labels[0]}")
plt.show()
```



4.3. 라이브러리 소개 및 활용 - Keras



- TensorFlow의 상위 API
 - 오픈 소스 신경망 라이브러리
 - 합성곱, 밀집 등 신경망 구성 설정 및 계산을 간단한 명령어로 쉽게 설계 가능
-
- data 전처리
 - : model.add(Conv2D(파라미터 값))
 - 합성곱 계산 및 기타 설정을 Conv2D() 하나로 쉽게 구성 가능
 - : .MaxPooling2D(2)
 - 2x2 size의 MaxPooling 적용
 - : .Flatten() - 다차원 data를 1차원으로 변환
 - : .Dropout('비율') - 비율만큼 출력값 상쇄
 - : 마지막 출력층 activation='softmax'
 - 5개 분류이므로 노드 5개중 가장 높은 value값을 예측값으로 출력

```
from tensorflow import keras
# 신경망 모델 구성
model = keras.Sequential()

model.add(keras.layers.Conv2D(128, kernel_size=3, activation='relu', padding='same', input_shape=(160, 160, 3)))
model.add(keras.layers.MaxPooling2D(2)) # 풀링 후 특성 맵 크기(80, 80, 128)

model.add(keras.layers.Flatten()) # 밀집층에 입력시키기 위해 3차원을 1차원으로 변환.

model.add(keras.layers.Dense(50, activation='relu'))
model.add(keras.layers.Dropout(0.2)) # 은닉층의 과대적합을 막기 위해 노드로 들어오는 입력을 랜덤으로 막음.

model.add(keras.layers.Dense(5, activation='softmax'))

# 신경망 모델 구조
model.summary()
```

```
Model: "sequential_2"

```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 160, 160, 128)	3584
max_pooling2d_2 (MaxPooling2D)	(None, 80, 80, 128)	0
flatten_2 (Flatten)	(None, 819200)	0
dense_4 (Dense)	(None, 50)	40960050
dropout_2 (Dropout)	(None, 50)	0
dense_5 (Dense)	(None, 5)	255

```

Total params: 40963889 (156.26 MB)
Trainable params: 40963889 (156.26 MB)
Non-trainable params: 0 (0.00 Byte)
```

5. Project 진행 과정

- 5.1 합성곱 층 필터 수 결정
- 5.2 밀집 층 노드 수 결정
- 5.3 합성곱 층 수 결정
- 5.4 밀집 층 수 결정
- 5.5 Hyperparameter 조정
- 5.6 최종 신경망

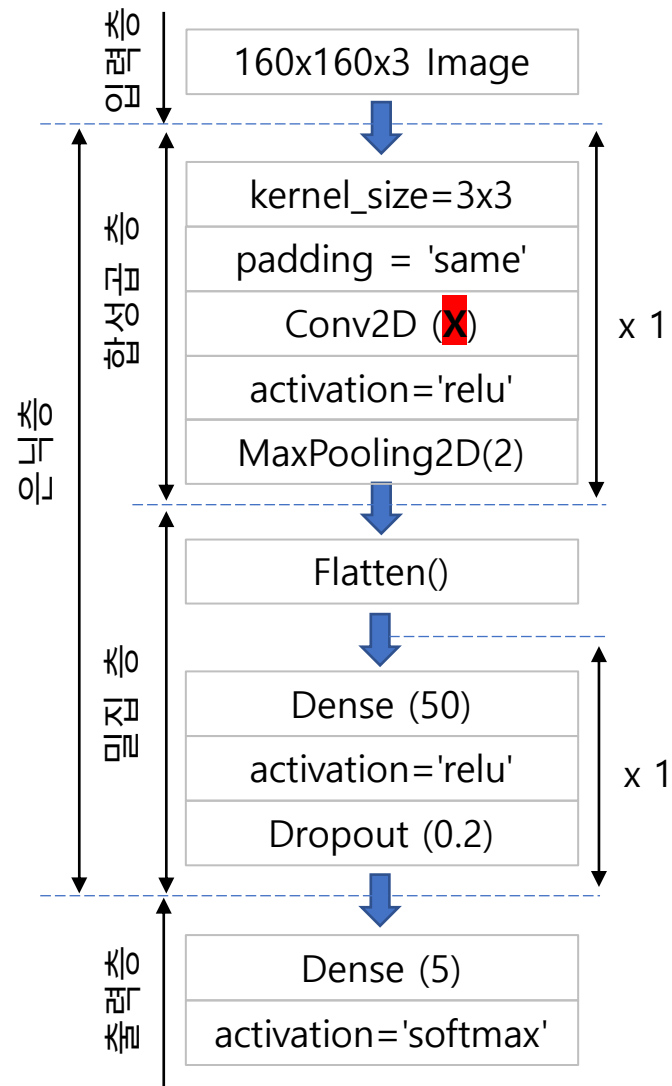
5.1 합성곱 층 필터 수 결정

Dataset 나누기

- 훈련 Data = $765 \times 5 = 3825$ 개
 - 검증 Data = $135 \times 5 = 675$ 개
 - 테스트 Data = $100 \times 5 = 500$ 개
- ▣ $3825 + 675 + 500 = 5000$ 개

과정	조작 변인	조작 내용	통제 변인
1	합성곱 층 필터 수	32 / 64 / 128 / 256	<ul style="list-style-type: none">통제 변인 값합성곱 층 수 = 1밀집 층 수 = 1밀집 층 노드 수 = 50
2	밀집 층 노드 수	25 / 50 / 100 / 200	<ul style="list-style-type: none">통제 변인 값합성곱 필터 수 = 1과정 결정합성곱 층 수 = 1밀집 층 수 = 1
3	합성곱 층 수	1 / 3 / 5	<ul style="list-style-type: none">통제 변인 값합성곱 필터 수 = 1과정 결정밀집 노드 수 = 2과정 결정밀집 층 수 = 1
4	밀집층 층 수	1 / 2 / 3	<ul style="list-style-type: none">통제 변인 값합성곱 필터 수 = 1과정 결정밀집 노드 수 = 2과정 결정합성곱 층 수 = 3과정 결정
5	은닉층의 필터, 노드 수와 층 수들이 정해져 대략적인 신경망 구조의 학습 결과를 본 후 Hyperparameter 값 조정(detail 조정)		

5.1 합성곱 층 필터 수 결정



<인공신경망 epoch = 15>

첫 epoch부터 과대적합 → 학습 Data 부족 → 이미지 증강 결정

loss - epoch graph		
X =	32	64
loss - epoch graph		
X =	128	256

5.1 합성곱 층 필터 수 결정 - 이미지 증강

ImageDataGenerator() - keras 제공

- Data 전처리 시 `resize(openCV)`, 배열 생성(`Numpy`), 정규화 (`Image Data/255`)를 각각 따로 했던 작업을 한꺼번에 설정 가능
- labeling : 폴더명 오름차순 순으로 자동으로 생성
(단, one-hot encoding 적용됨 => `loss='categorical_crossentropy'`)
- `ImageDataGenerator()` 객체를 생성 시, 이미지 증강 관련 설정 : 설정 값은 신경망 모델 학습 시 한 epoch 당 하나의 Image가 설정 값 수만큼 증강
 - ① `rotation_range = 180` : 0~180도 무작위 회전
 - ② `width_shift_range = 0.4` : 가로 방향으로 40% 내 무작위 이동
 - ③ `height_shift_range = 0.4` : 세로 방향으로 40% 내 무작위 이동
 - ④ `shear_range = 0.4` : 임의 축으로 40% 내 무작위 기울임
 - ⑥ `zoom_range = 0.4` : 40% 무작위 zoom in/out
 - ⑦ `horizontal_flip = True` : 좌우 반전
 - ⑧ `vertical_flip = True` : 상하 반전: 증강 관련 8가지 옵션 설정
data 수 = 원본 data 수 x 옵션 수 x epoch 수
data 수 = $3825 \times 8 \times 15 = 459000$ 개 data 생성

```
from keras.preprocessing.image import ImageDataGenerator
# ImageDataGenerator 객체 생성
train_data_gen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 180,
    width_shift_range = 0.4,
    height_shift_range = 0.4,
    shear_range = 0.4,
    zoom_range = 0.4,
    horizontal_flip = True,
    vertical_flip = True,
    fill_mode = 'nearest'
)
# dir 순회하며 dir명에 맞게 라벨 생성하는 반복자 객체 생성
# 각각의 반복자 객체에는 dir의 이미지들이 랜덤순으로 들어감.
train_gen = train_data_gen.flow_from_directory(
    train_dir,
    target_size = (160,160),
    class_mode = 'categorical'
)
```


5.1 합성곱 층 필터 수 결정 - 이미지 증강



원본 380x290

rotation
zoom out



증강 160x160



원본 1424x2144

rotation
zoom in



증강 160x160



원본 1000x904

vertical_flip
with_shift



증강 160x160



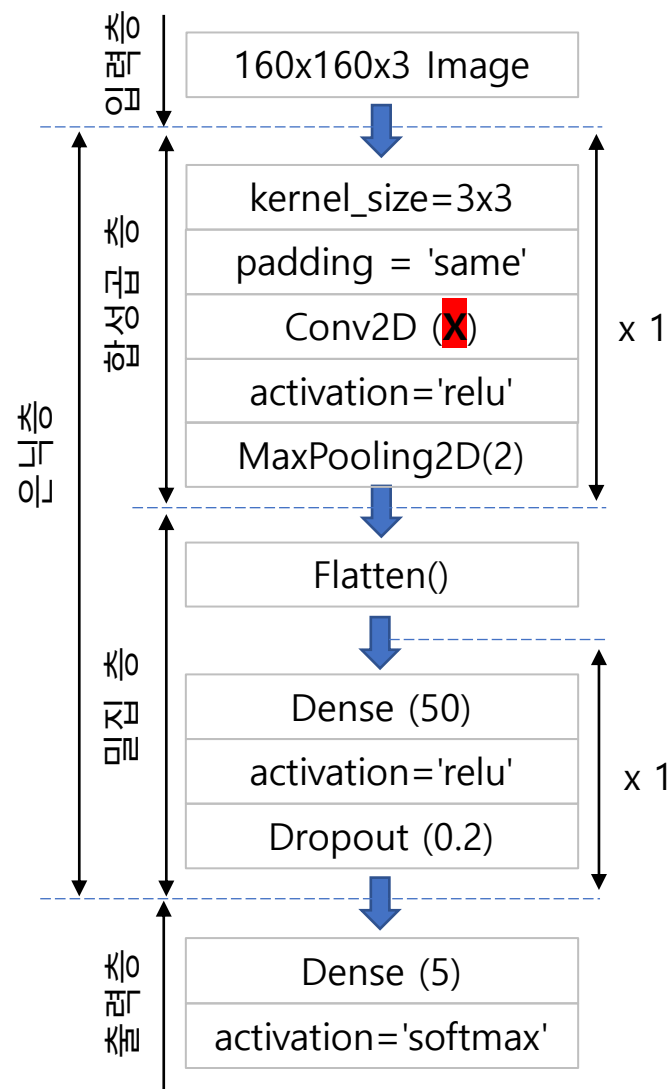
원본 300x300

horizontal_flip
zoom in



증강 160x160

5.1 합성곱 층 필터 수 결정 - 이미지 증강

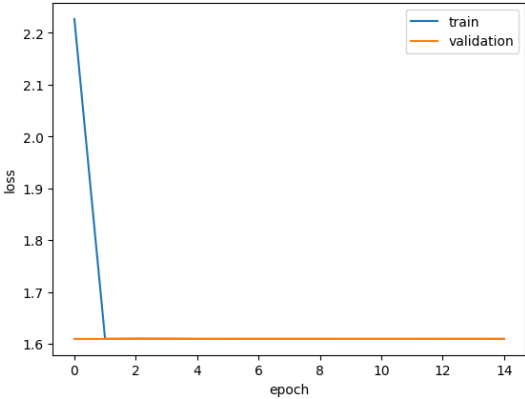
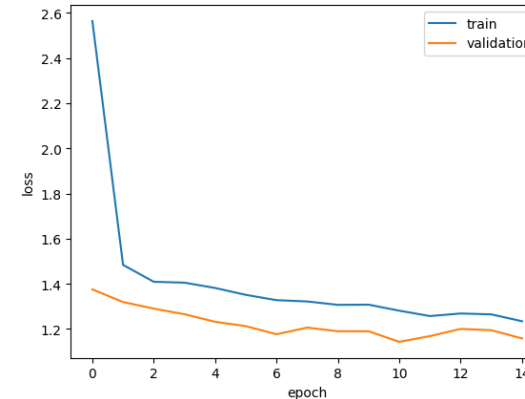
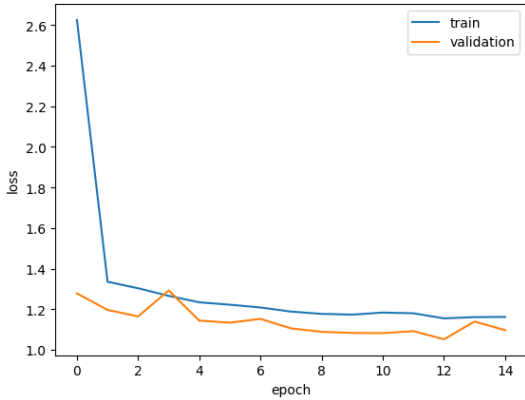
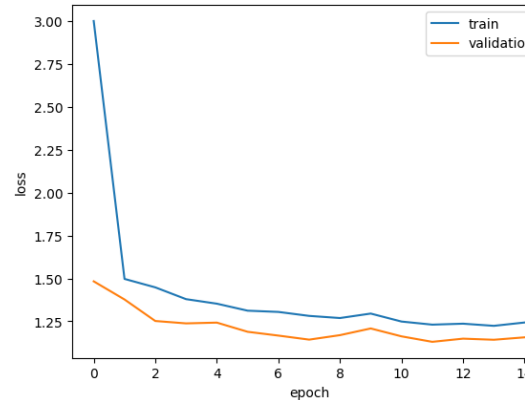


<인공신경망 epoch = 15>

loss graph
유사

Test data
평가 결과 확인

32	loss : 1.6094 acc : 0.2000	64	loss : 1.1190 acc : 0.5420
128	loss : 1.0807 acc : 0.5740	256	loss : 1.1383 acc : 0.5400

loss - epoch graph		
X =	32	64
loss - epoch graph		
X =	128	256

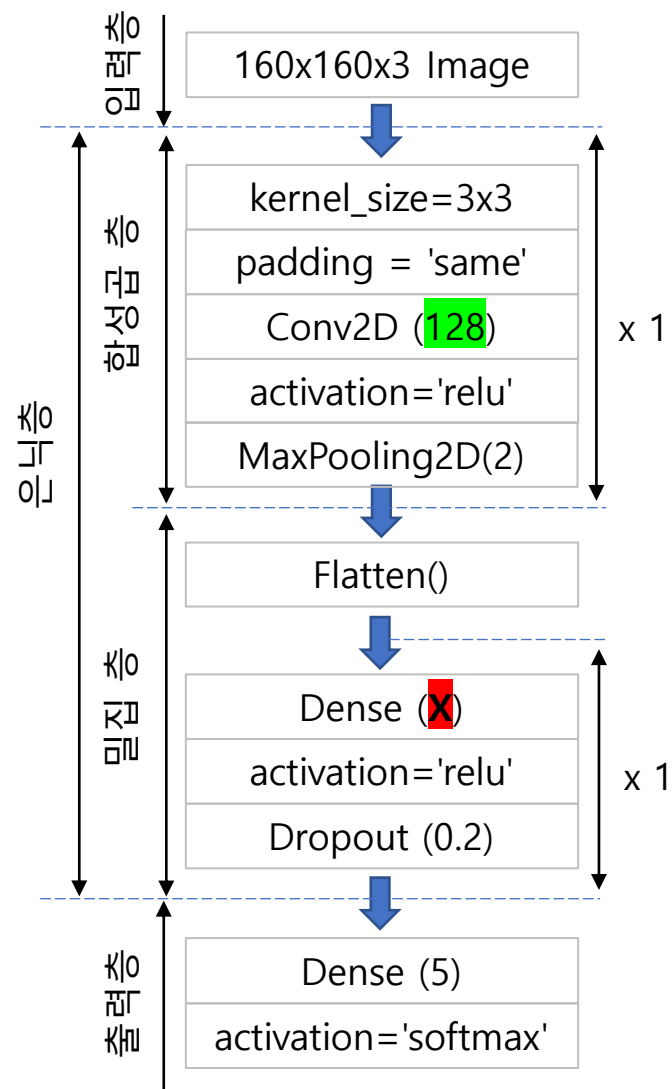
5.2 밀집 층 노드 수 결정 - 이미지 증강

Dataset 나누기

- 훈련 Data = $765 \times 5 = 3825$ 개
- 검증 Data = $135 \times 5 = 675$ 개
- 테스트 Data = $100 \times 5 = 500$ 개
- ▣ $3825 + 675 + 500 = 5000$ 개

과정	조작 변인	조작 내용	통제 변인
1	합성곱 층 필터 수	32 / 64 / 128 / 256	<ul style="list-style-type: none"> · 통제 변인 값 · 합성곱 층 수 = 1 · 밀집 층 수 = 1 · 밀집 층 노드 수 = 50
2	밀집 층 노드 수	25 / 50 / 100 / 200	<ul style="list-style-type: none"> · 통제 변인 값 · 합성곱 필터 수 = 1과정 결정 · 합성곱 층 수 = 1 · 밀집 층 수 = 1
3	합성곱 층 수	1 / 3 / 5	<ul style="list-style-type: none"> · 통제 변인 값 · 합성곱 필터 수 = 1과정 결정 · 밀집 노드 수 = 2과정 결정 · 밀집 층 수 = 1
4	밀집층 층 수	1 / 2 / 3	<ul style="list-style-type: none"> · 통제 변인 값 · 합성곱 필터 수 = 1과정 결정 · 밀집 노드 수 = 2과정 결정 · 합성곱 층 수 = 3과정 결정
5	은닉층의 필터, 노드 수와 층 수들이 정해져 대략적인 신경망 구조의 학습 결과를 본 후 Hyperparameter 값 조정(detail 조정)		

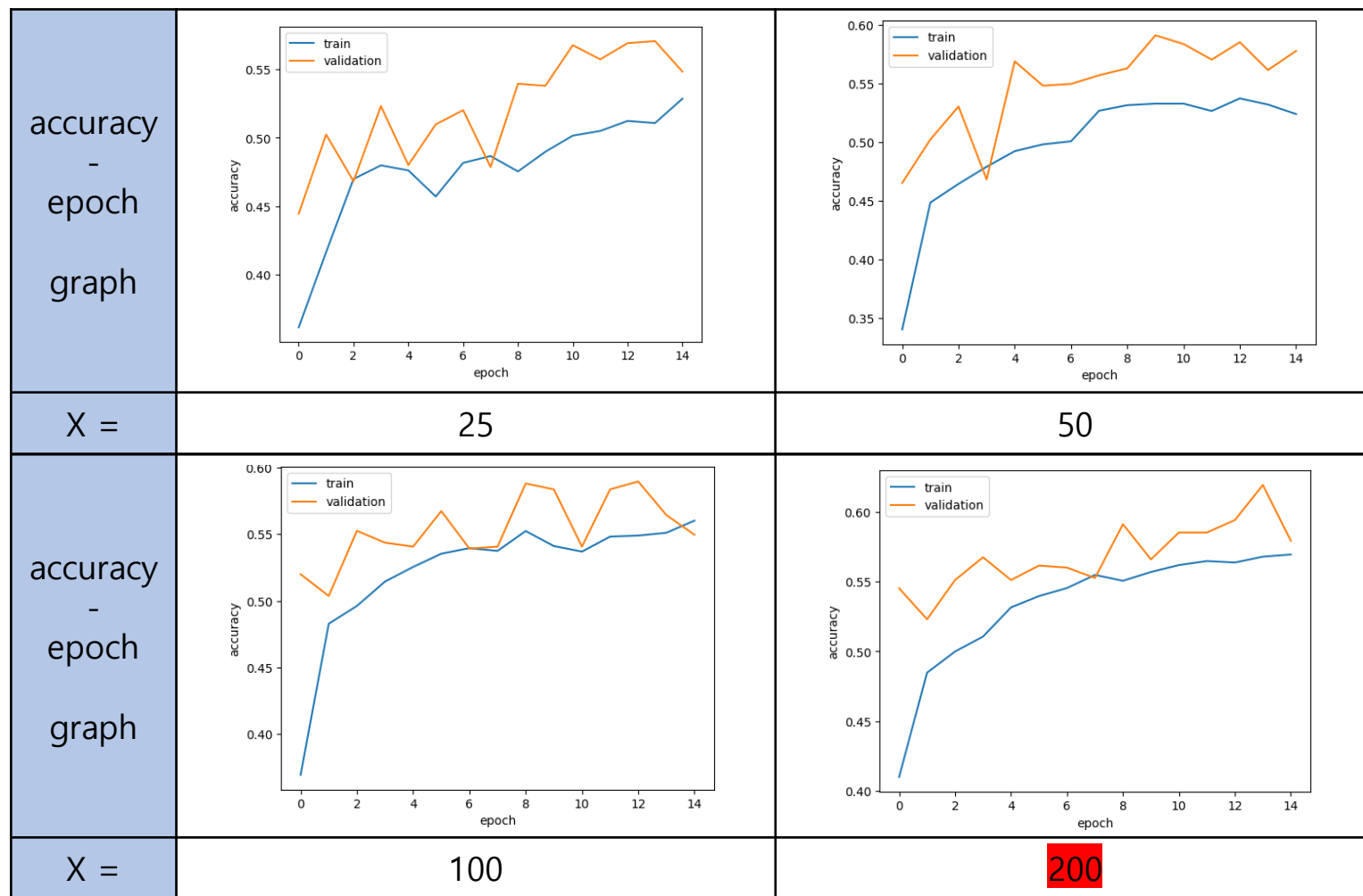
5.2 밀집 층 노드 수 결정 - 이미지 증강



loss graph
유사

Test data
평가 결과 확인

25	loss : 1.0942 acc : 0.5600	50	loss : 1.0807 acc : 0.5740
100	loss : 1.0812 acc : 0.5640	200	loss : 1.0356 acc : 0.5660



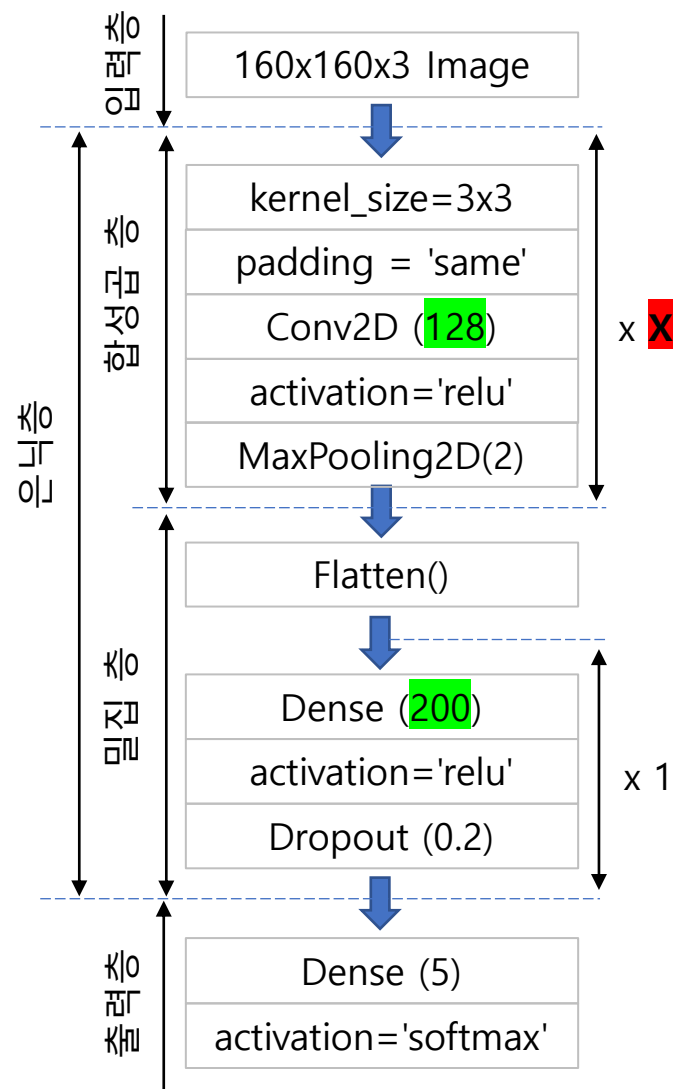
5.3 합성곱 층 수 결정 - 이미지 증강

Dataset 나누기

- 훈련 Data = $765 \times 5 = 3825$ 개
- 검증 Data = $135 \times 5 = 675$ 개
- 테스트 Data = $100 \times 5 = 500$ 개
- ▣ $3825 + 675 + 500 = 5000$ 개

과정	조작 변인	조작 내용	통제 변인
1	합성곱 층 필터 수	32 / 64 / 128 / 256	<ul style="list-style-type: none"> 통제 변인 값 합성곱 층 수 = 1 밀집 층 수 = 1 밀집 층 노드 수 = 50
2	밀집 층 노드 수	25 / 50 / 100 / 200	<ul style="list-style-type: none"> 통제 변인 값 합성곱 필터 수 = 1과정 결정 합성곱 층 수 = 1 밀집 층 수 = 1
3	합성곱 층 수	1 / 3 / 5	<ul style="list-style-type: none"> 통제 변인 값 합성곱 필터 수 = 1과정 결정 밀집 노드 수 = 2과정 결정 밀집 층 수 = 1
4	밀집층 층 수	1 / 2 / 3	<ul style="list-style-type: none"> 통제 변인 값 합성곱 필터 수 = 1과정 결정 밀집 노드 수 = 2과정 결정 합성곱 층 수 = 3과정 결정
5	은닉층의 필터, 노드 수와 층 수들이 정해져 대략적인 신경망 구조의 학습 결과를 본 후 Hyperparameter 값 조정(detail 조정)		

5.3 합성곱 층 수 결정 - 이미지 증강



<인공신경망 epoch = 15>

loss graph
유사

Test data
평가 결과 확인

1	loss : 1.0356 acc : 0.5660	-	-
3	loss : 0.9709 acc : 0.6160	5	loss : 0.9363 acc : 0.6340

accuracy - epoch graph		-
X =	1	-
accuracy - epoch graph		
X =	3	5

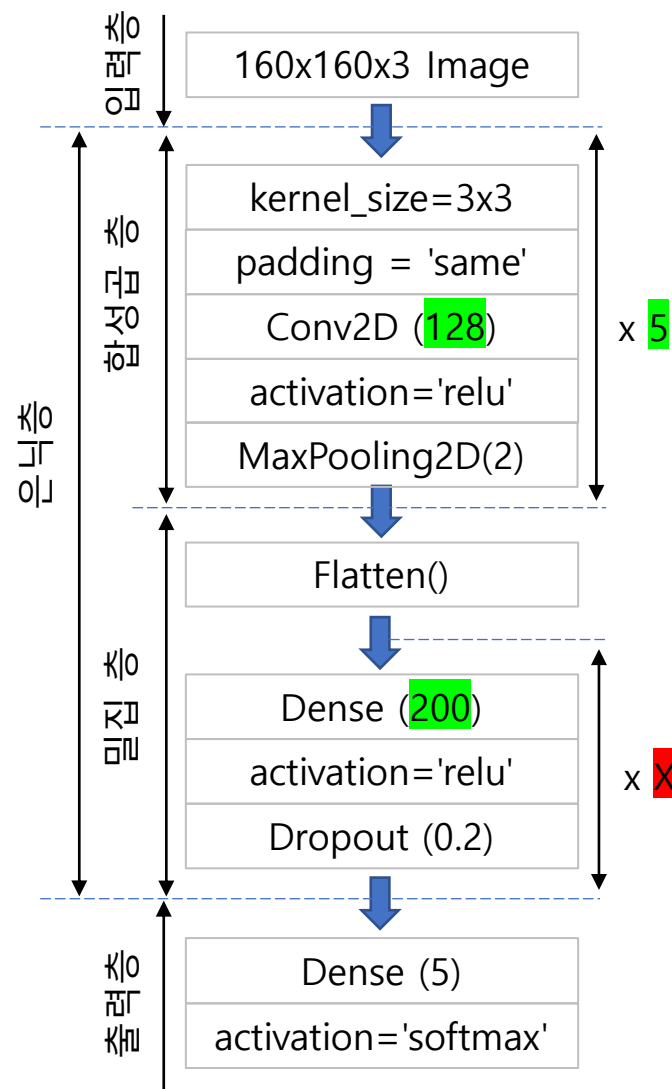
5.4 밀집 층 수 결정 - 이미지 증강

Dataset 나누기

- 훈련 Data = $765 \times 5 = 3825$ 개
- 검증 Data = $135 \times 5 = 675$ 개
- 테스트 Data = $100 \times 5 = 500$ 개
- ▣ $3825 + 675 + 500 = 5000$ 개

과정	조작 변인	조작 내용	통제 변인
1	합성곱 층 필터 수	32 / 64 / 128 / 256	<ul style="list-style-type: none"> 통제 변인 값 합성곱 층 수 = 1 밀집 층 수 = 1 밀집 층 노드 수 = 50
2	밀집 층 노드 수	25 / 50 / 100 / 200	<ul style="list-style-type: none"> 통제 변인 값 합성곱 필터 수 = 1과정 결정 합성곱 층 수 = 1 밀집 층 수 = 1
3	합성곱 층 수	1 / 3 / 5	<ul style="list-style-type: none"> 통제 변인 값 합성곱 필터 수 = 1과정 결정 밀집 노드 수 = 2과정 결정 밀집 층 수 = 1
4	밀집 층 수	1 / 2 / 3	<ul style="list-style-type: none"> 통제 변인 값 합성곱 필터 수 = 1과정 결정 밀집 노드 수 = 2과정 결정 합성곱 층 수 = 3과정 결정
5	은닉층의 필터, 노드 수와 층 수들이 정해져 대략적인 신경망 구조의 학습 결과를 본 후 Hyperparameter 값 조정(detail 조정)		

5.4 밀집 층 수 결정 - 이미지 증강



<인공신경망 epoch = 15>

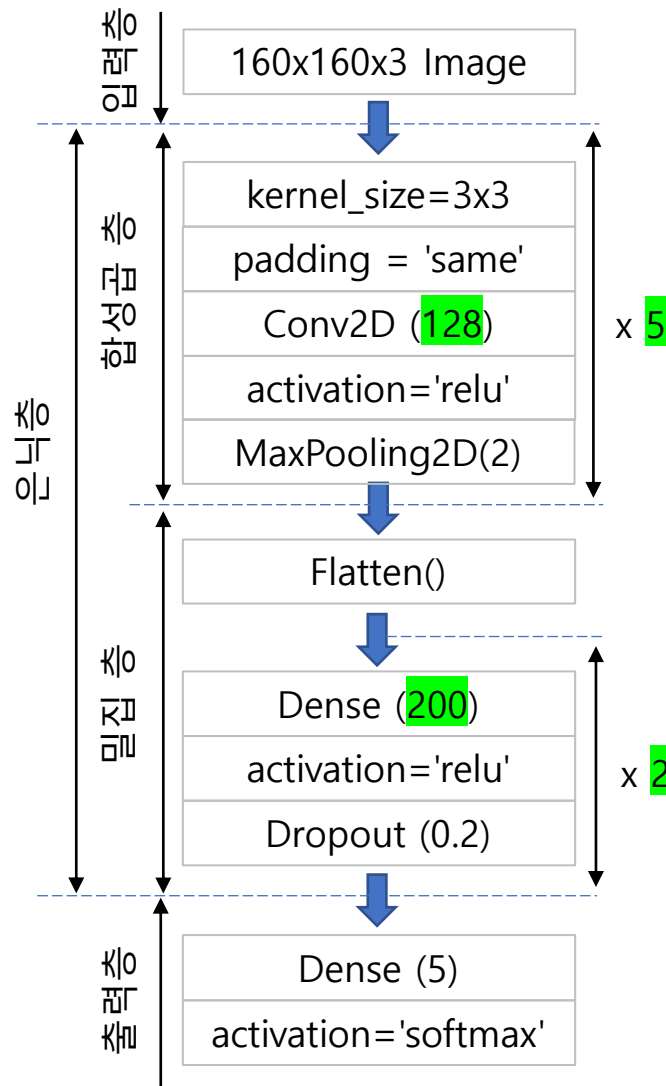
loss graph
유사

Test data
평가 결과 확인

1	loss : 0.9709 acc : 0.6160	-	-
2	loss : 0.9039 acc : 0.6360	3	loss : 0.9731 acc : 0.6100

accuracy - epoch graph		-
X =	1	-
accuracy - epoch graph		
X =	2	3

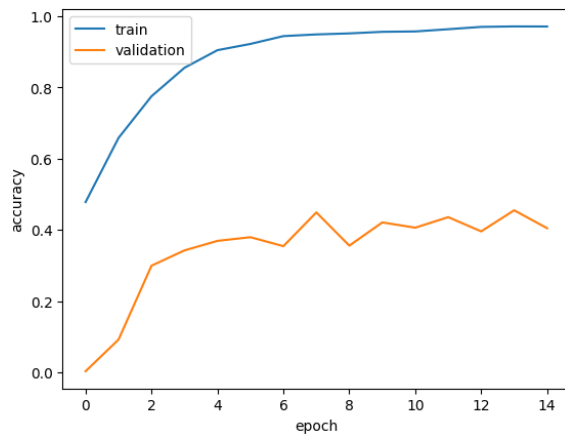
5.1~5.4 종합 - 이미지 증강



<인공신경망 epoch = 15>

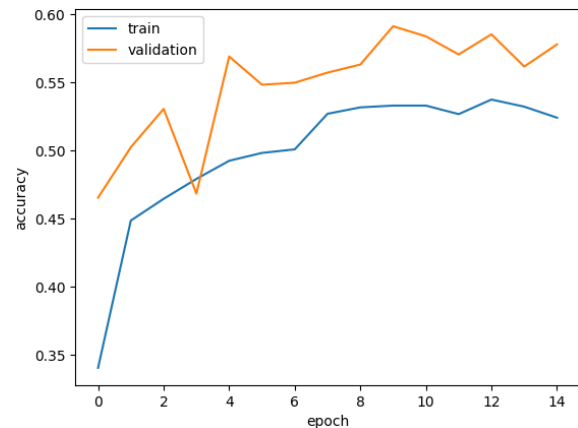
종합(1st. 기준)

- loss : - 0.5531
(1.4570->0.9039)
- accuracy : -0.144
(0.7800->0.6360)
- ★ epoch 수는 동일한데 학습 데이터 수는 늘어났기 때문에 정확도가 떨어졌다고 판단 => epoch 수 늘리기

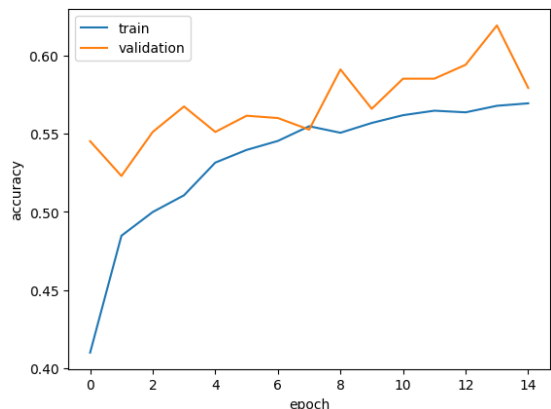


<합성곱 층 1-128, 밀집 층 1-50>
<loss : 1.4570, accuracy : 0.7800>

→
이미지
증강

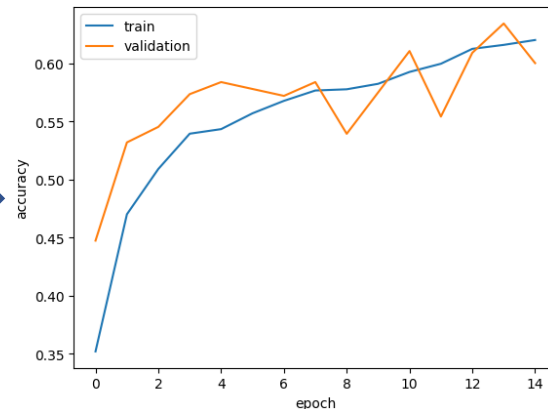


<합성곱 층 1-128, 밀집 층 1-50>
<loss : 1.0807, accuracy : 0.5740>



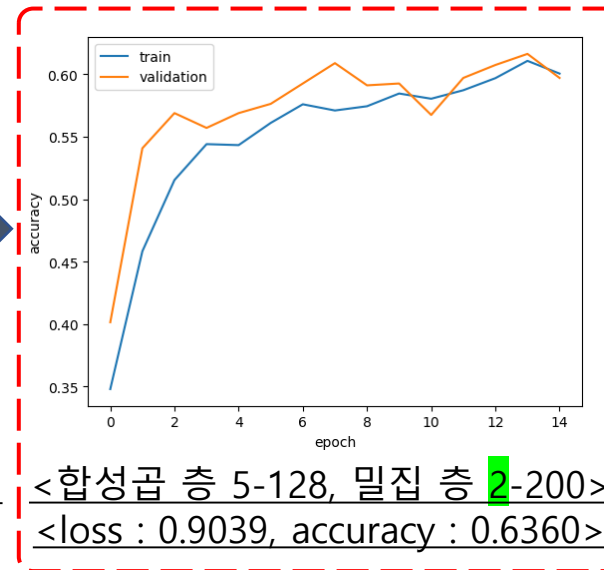
<합성곱 층 1-128, 밀집 층 1-200>
<loss : 1.0356, accuracy : 0.5660>

→



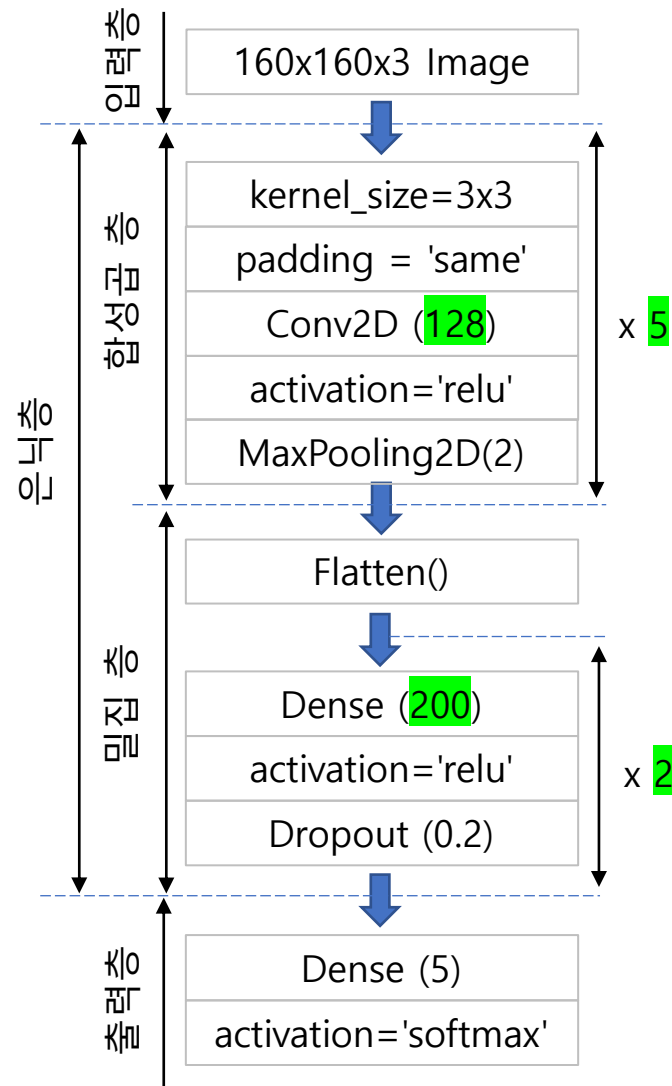
<합성곱 층 5-128, 밀집 층 1-200>
<loss : 0.9363, accuracy : 0.6340>

→



<합성곱 층 5-128, 밀집 층 2-200>
<loss : 0.9039, accuracy : 0.6360>

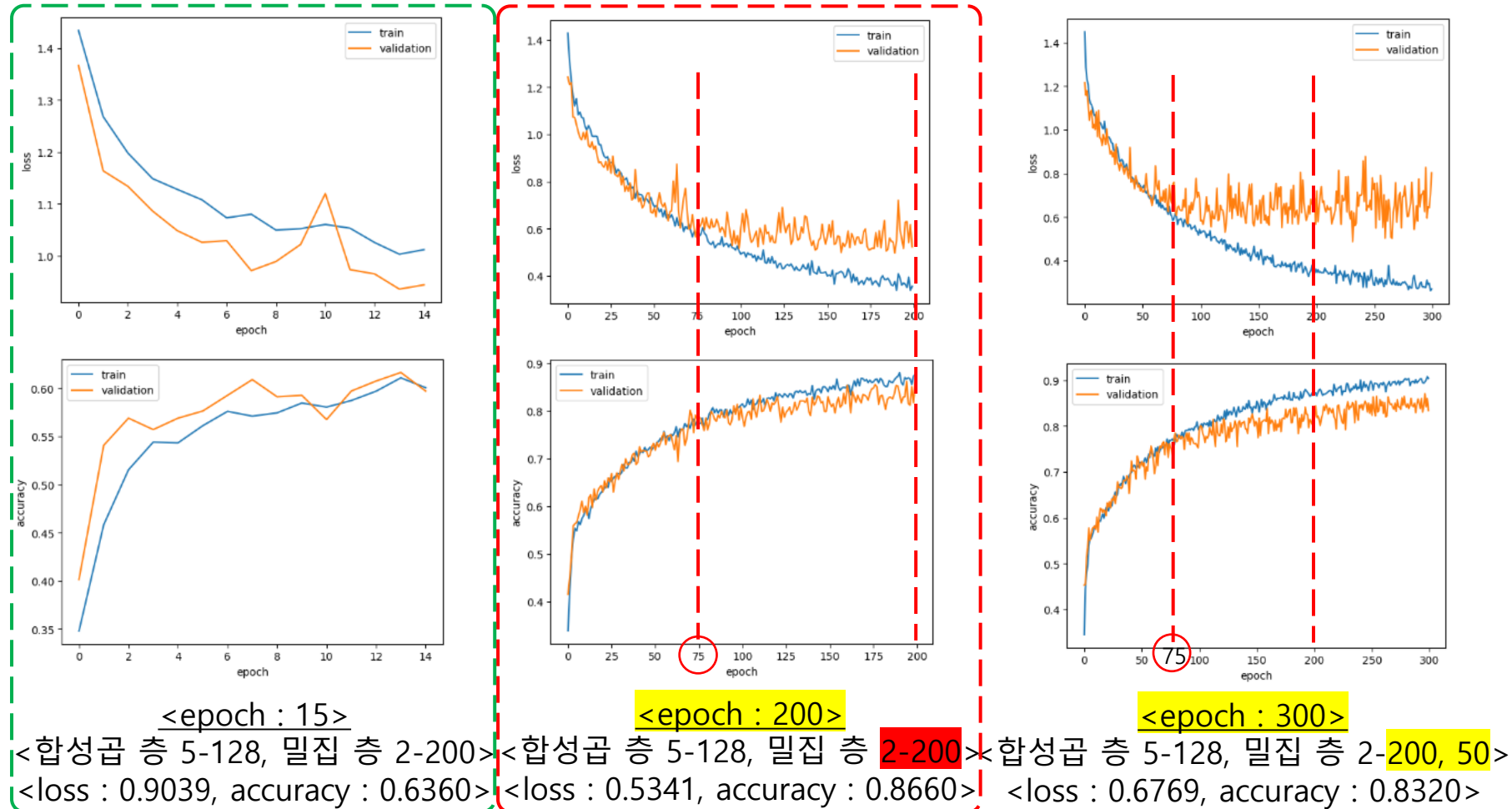
5.5 Hyperparameter 조정(epoch) - 이미지 증강



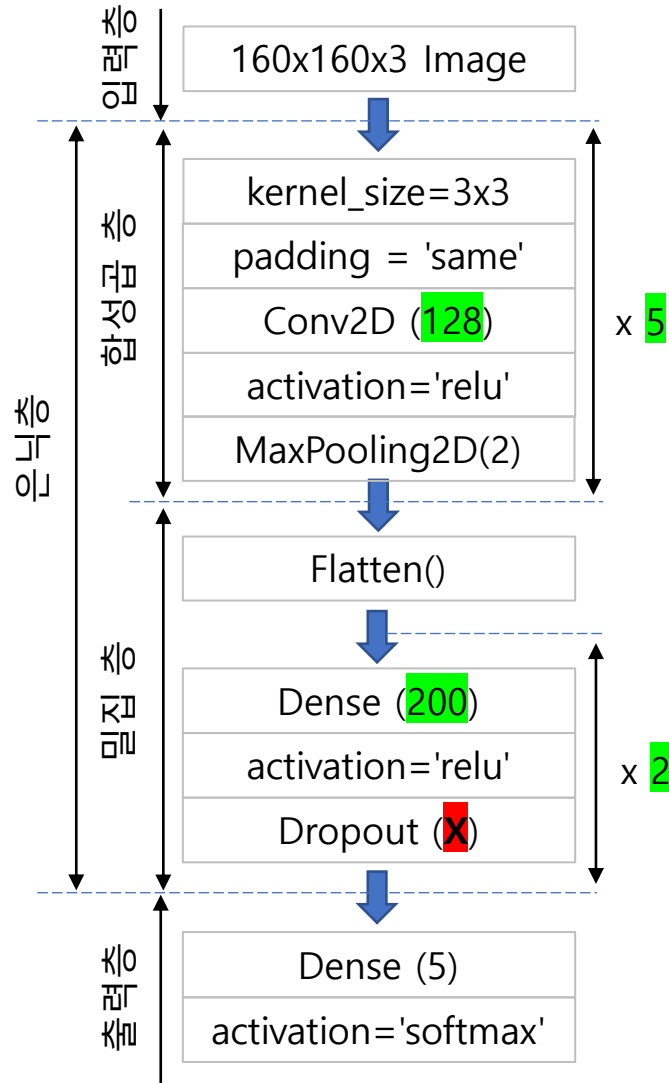
<인공신경망 epoch = **X**>

고찰

- epoch값 늘려 더 많은 증강 data 생성 => loss, accuracy 향상
- 밀집 층 노드 수 달리함 => loss, accuracy 저하 => 밀집 층 노드 수 유지 결정



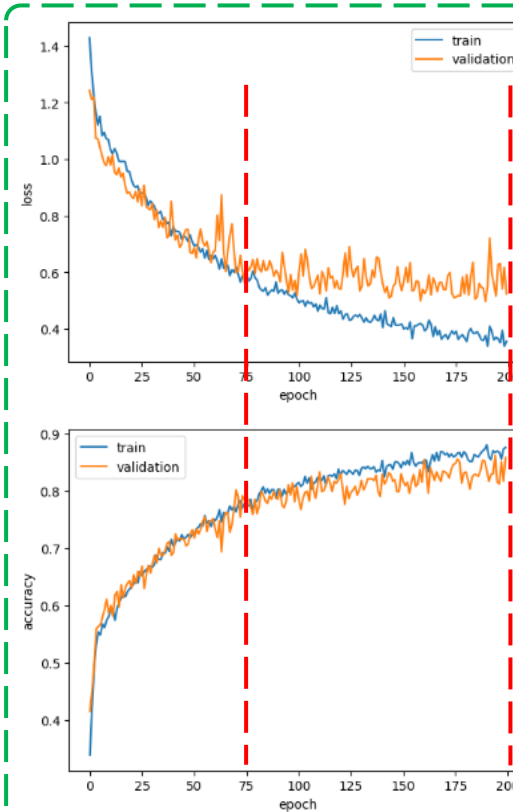
5.5 Hyperparameter 조정(dropout) - 이미지 증강



<인공신경망 epoch = 300>

고찰

- epoch 75 즈음부터 시작되는 과대적합 해소 위해 dropout 0.4 적용 => 유의미한 변화 없음
- 다른 Hyperparameter 변경에 영향력이 있을 것이란 생각에 밀집 층 2개 중 하나 0.2, 다른 하나 0.4 적용한 신경망 채택

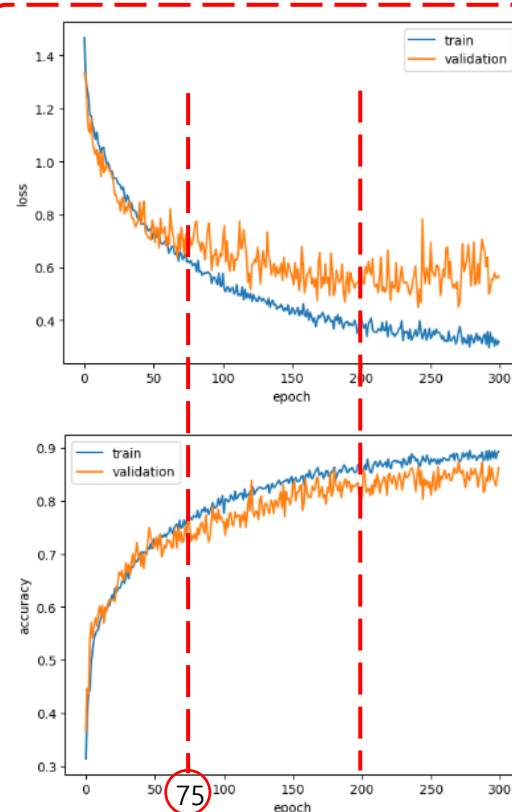


<epoch : 200>

<합성곱 층 5-128, 밀집 층 2-200>

<밀집 층 dropout : 0.2>

<loss : 0.5341, accuracy : 0.8660>

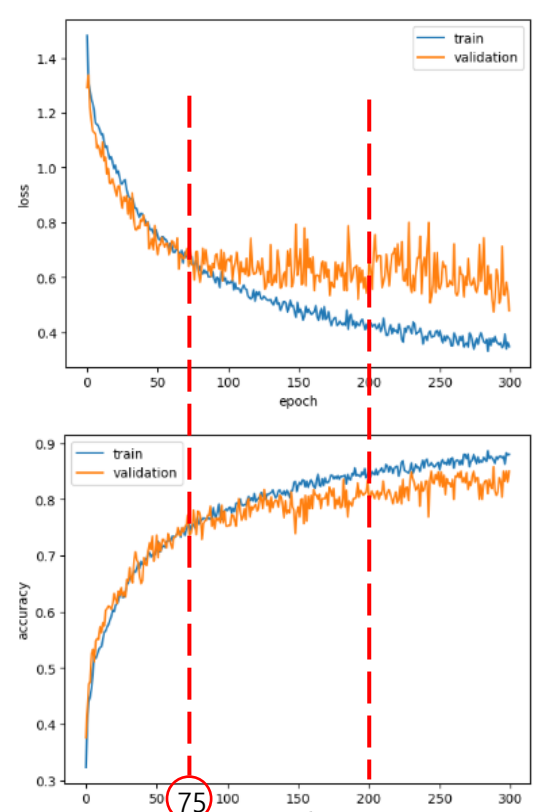


<epoch : 300>

<합성곱 층 5-128, 밀집 층 2-200>

<밀집 층 dropout : 0.2, 0.4>

<loss : 0.5542, accuracy : 0.8720>



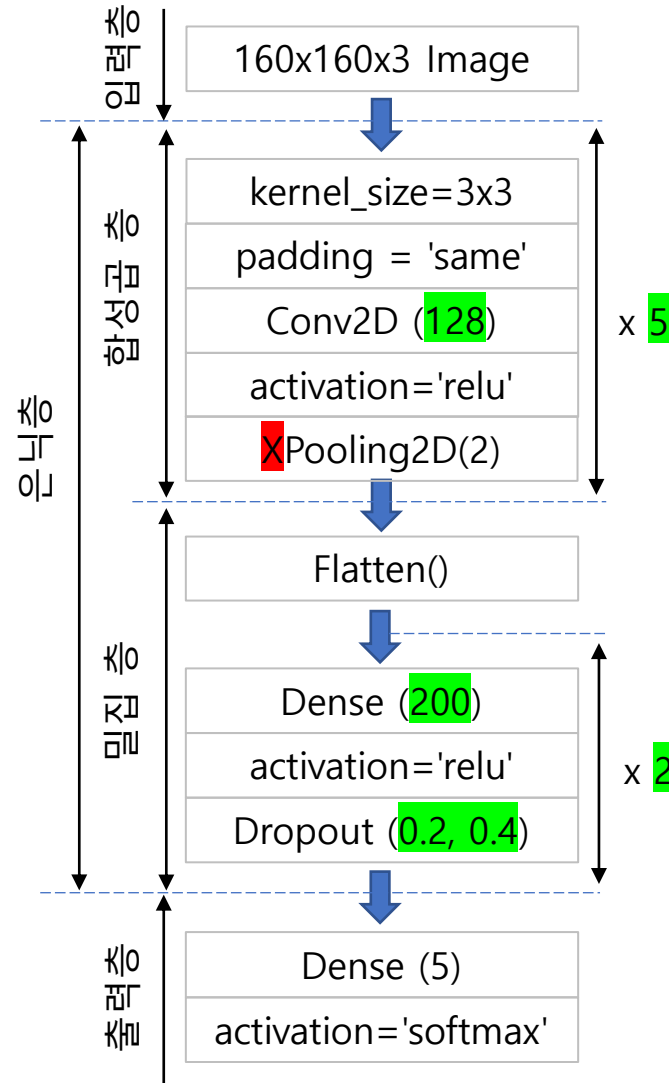
<epoch : 300>

<합성곱 층 5-128, 밀집 층 2-200>

<밀집 층 dropout : 0.4>

<loss : 0.5820, accuracy : 0.8360>

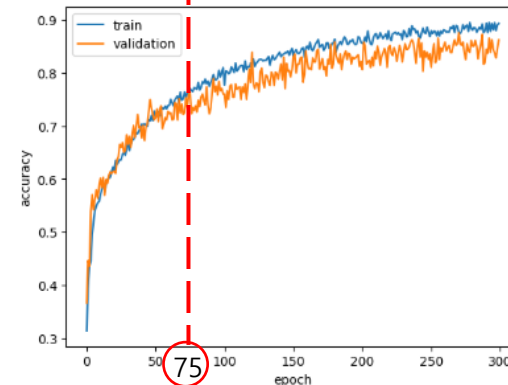
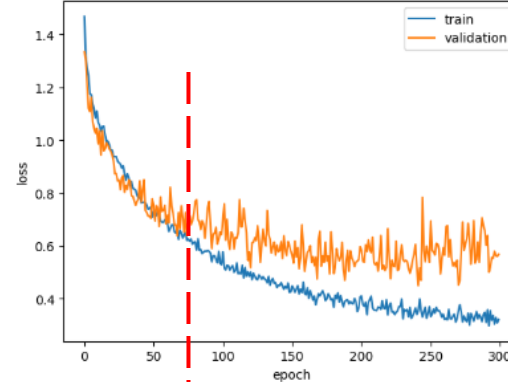
5.5 Hyperparameter 조정(Pooling) - 이미지 증강



<인공신경망 epoch = 300>

고찰

- 과대적합 특징을 너무 잘 잡나? => Max 아닌 AveragePooling 적용
=> 여전히 epoch75 즈음에서 과대적합 시작
- loss값은 약간 저하되었지만 정확도 향상

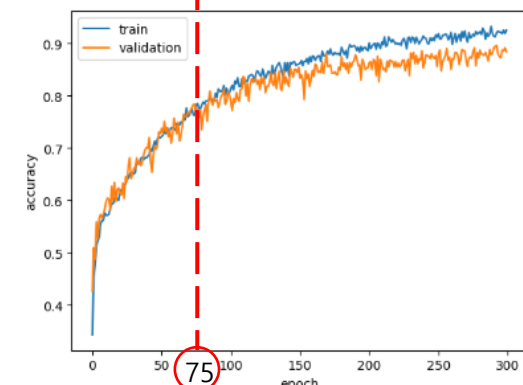
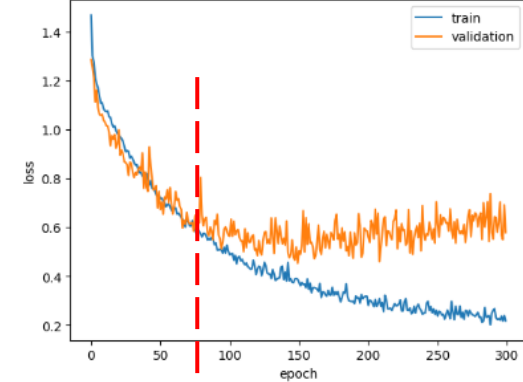


<epoch : 300>

<합성곱 층 5-128, 밀집 층 2-200>

<합성곱 층 MaxPooling2D(2)>

<loss : 0.5542, accuracy : 0.8720>



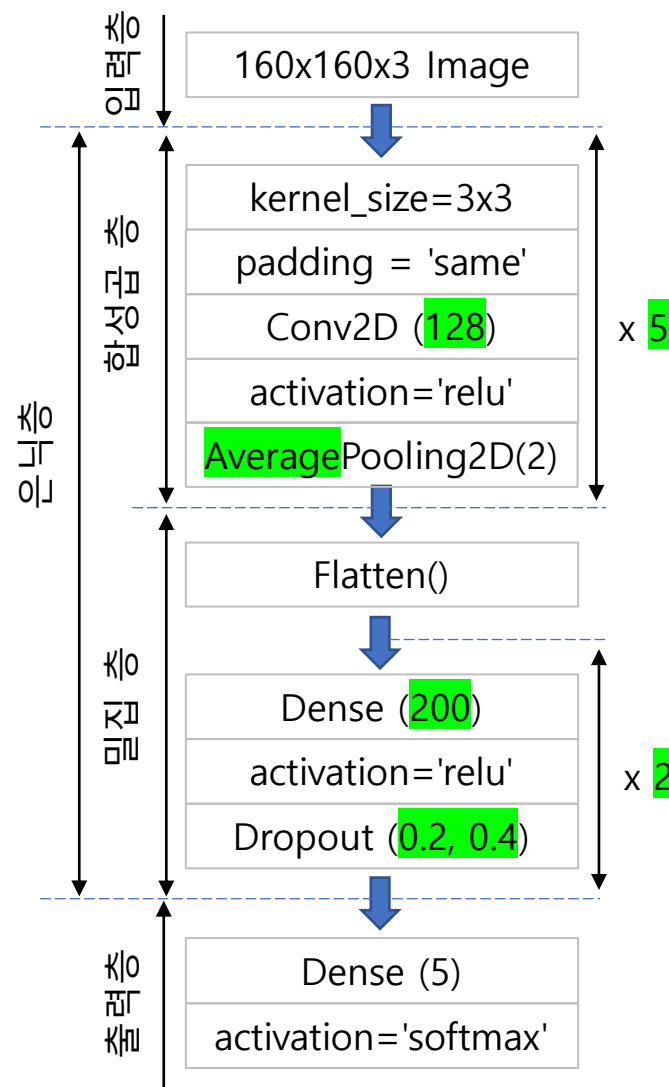
<epoch : 300>

<합성곱 층 5-128, 밀집 층 2-200>

<합성곱 층 Averagepooling2D(2)>

<loss : 0.5793, accuracy : 0.9160>

5.6 최종 신경망 - 이미지 증강



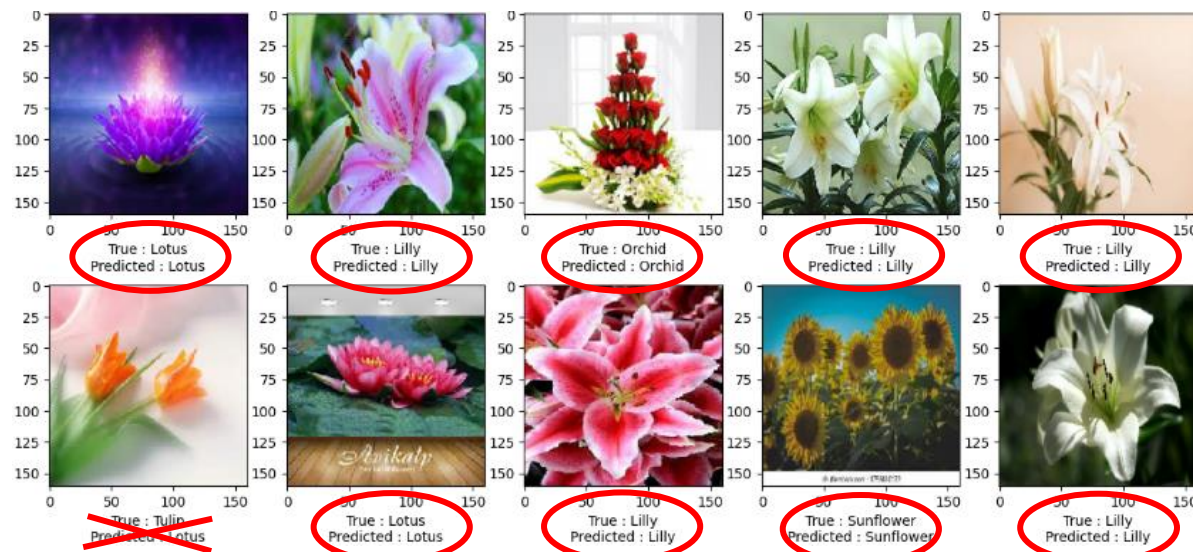
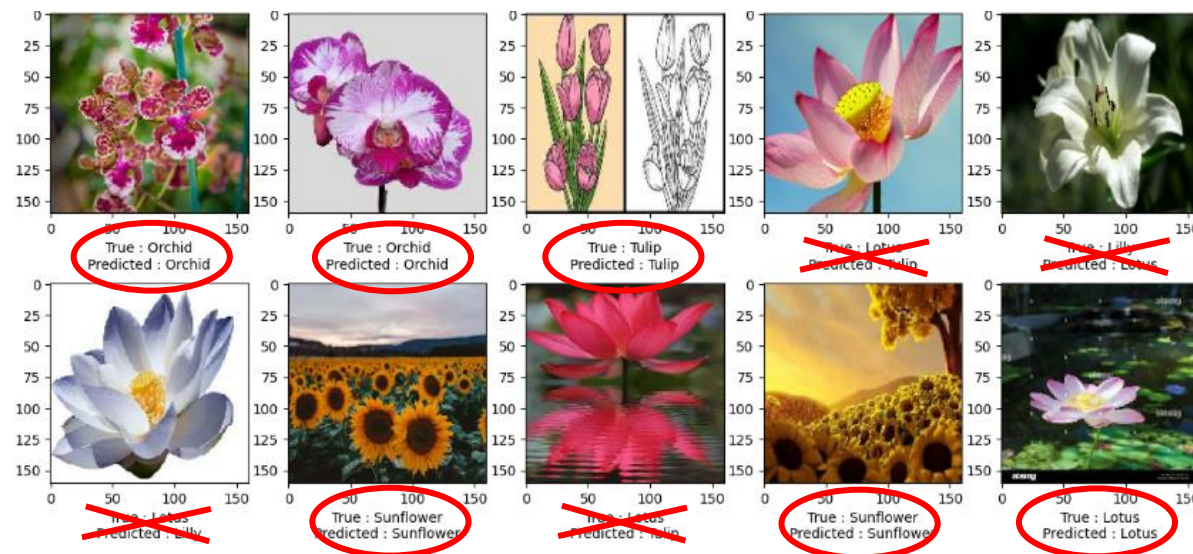
<인공신경망 epoch = 300>

초기 신경망

- Conv2D 1-128
- MaxPooling2D(2)
- Dense 1-50
- Dropout(0.2)
- loss : 1.0807
- accuracy : 0.5740

최종 신경망

- Conv2D 5-128
- AveragePooling2D(2)
- Dense 2-200
- Dropout(0.2, 0.4)
- loss : 0.5793
- accuracy : 0.9160



6. 결과 및 고찰

6. 결과 및 고찰

- 합성곱 층 필터 수 : 많다고 좋은 것이 아님.

32	loss : 1.6094 acc : 0.2000	64	loss : 1.1190 acc : 0.5420
128	loss : 1.0807 acc : 0.5740	256	loss : 1.1383 acc : 0.5400

- 밀집 층 노드 수 : 많을 수록 좋음.

25	loss : 1.0942 acc : 0.5600	50	loss : 1.0807 acc : 0.5740
100	loss : 1.0812 acc : 0.5640	200	loss : 1.0356 acc : 0.5660

- 합성곱 층 수 : 많을 수록 좋음.

1	loss : 1.0356 acc : 0.5660	-	-
3	loss : 0.9709 acc : 0.6160	5	loss : 0.9363 acc : 0.6340

- 밀집 층 수 : 많다고 좋은 것이 아님.

1	loss : 0.9709 acc : 0.6160	-	-
2	loss : 0.9039 acc : 0.6360	3	loss : 0.9731 acc : 0.6100

고찰

최종 신경망에서 정확도 91.6%의 성능을 내어 '90% 이상 정확도를 가진 신경망 설계'라는 이번 프로젝트의 목표는 달성하였다.

인공신경망 구성에서 가장 중요한 것은 은닉층의 층 수와 노드 수라고 생각하였다. 따라서 이 수들을 결정하기 위해 Pooling, Dropout, 이미지 크기 등 많은 값들을 고정시킬 필요가 있었다. 특히 Pooling을 합성곱 한층마다 적용하니 층 수 늘리는데 한계가 있었다(160, 80, 40, 20, 10, 5=최대 5층). 알아내고자 한 층 수 하나만을 변수로 두고 변수에 따른 신경망 성능이 유의미하게 차이가 나지 않아 각각의 수를 결정할 때 완전한 확신 없이 결정한 부분이 아쉬운 점이다. 하지만, 노드 수나, 필터 수 모두 어느 값까지는 많을 수록 신경망 성능이 좋아진다는 결론은 얻을 수 있었다.

신경망 구성은 parameter로 지정 가능한 수가 무한이므로 신경망의 수 또한 무한일 것이다. 이 중 많은 수를 한 값으로 가정하며 차근차근 찾기를 반복해야 좋은 성능을 내는 신경망을 찾을 수 있다고 생각한다. 따라서 최고의 신경망을 찾기는 불가능하지만 목표로 하는 신경망은 구성할 수 있을 것이다. 이번 프로젝트를 진행하며 뒤로 갈수록 층 수, 노드 수가 늘어나 하나의 신경망을 학습시키면 i7-6세대, 16GB RAM 기준 하루 넘게 걸렸다. 하지만 구독형 코랩 T4-GPU 기준으로 3시간 정도 걸렸다. 학습 속도에는 컴퓨터 하드웨어도 큰 영향력이 미친다는 것을 알 수 있었다.

감사합니다.