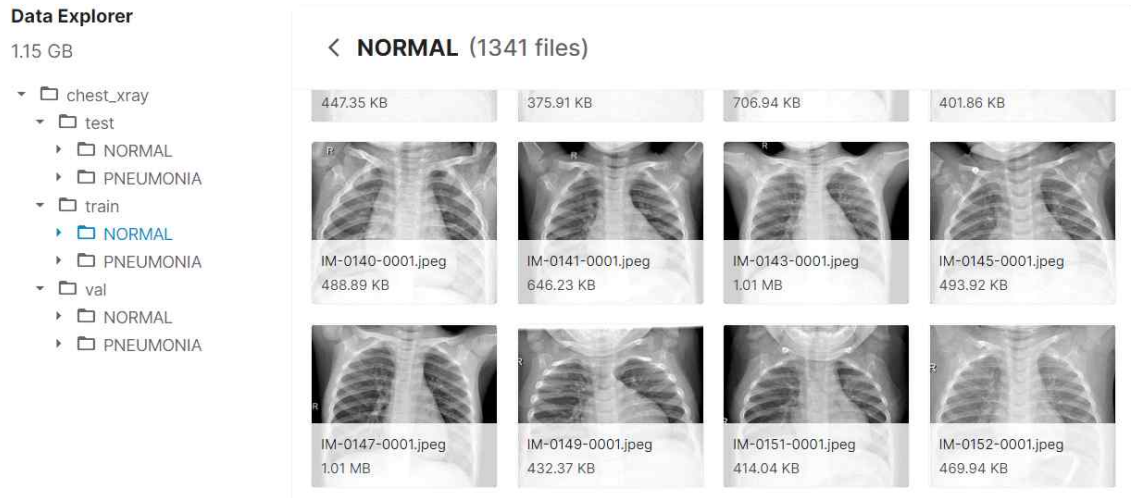
과목명	딥러닝/클라우드
담당교수	오세종 교수님
학과	소프트웨어학과
학번	32153180
이름	이상민
제출일자	2020.12.12

흉부 X-ray 사진으로 폐렴 진단 모델 작성하기

■ Dataset

- Chest X-Ray Images (Pneumonia)
  (https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia)



■ Environment

- Google colab
- Tensorflow 2.2.0

■ Setting

- Google colab

```
from google.colab import drive
drive.mount('/gdrive')

!pip install tensorflow==2.2.0
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

%cd /gdrive/MyDrive/DeepLearning/chest_xray/
%ls
```

```
/gdrive/MyDrive/DeepLearning/chest_xray
chest_xray/  __MACOSX/  test/  train/  val/
```

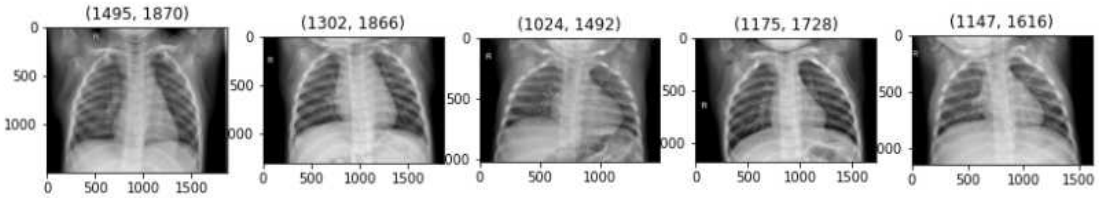
- Library

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import BatchNormalization
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Flatten, Dense, Dropout, Activation
from keras.preprocessing.image import ImageDataGenerator
```

■ Introduction

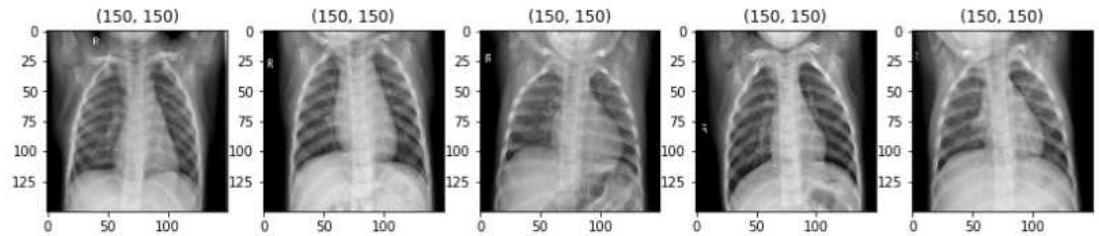
- Data shape

```
train_path = 'train'
test_path = 'test'
val_path = 'val'
labels = ['NORMAL', 'PNEUMONIA']

plt.figure(figsize=(14, 10))
for label in labels:
  path = os.path.join(train_path, label)

  idx = 1
  for image in os.listdir(path):
    img = cv2.imread(os.path.join(path, image), cv2.IMREAD_GRAYSCALE)
    plt.subplot(1, 5, idx)
    plt.title(img.shape)
    plt.imshow(img, cmap='gray')
    idx += 1

    if idx == 6:
      break
  break
plt.show()
```



- Resize example

```
plt.figure(figsize=(14, 10))
for label in labels:
  path = os.path.join(train_path, label)

  idx = 1
  for image in os.listdir(path):
    img = cv2.imread(os.path.join(path, image), cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (150, 150))
    plt.subplot(1, 5, idx)
    plt.title(img.shape)
    plt.imshow(img, cmap='gray')
    idx += 1

    if idx == 6:
      break
  break
plt.show()
```



이미지 데이터 크기가 모두 달라 (150, 150) 크기로 조정해서 사용했다.

■ Prepare data

- ImageDataGenerator 이용

```
datagen = ImageDataGenerator(rescale=1./255,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True)
train_data = datagen.flow_from_directory('train',
                                         target_size=(150, 150),
                                         batch_size=32,
                                         color_mode='grayscale',
                                         class_mode='binary')
val_data = datagen.flow_from_directory('val',
                                       target_size=(150, 150),
                                       batch_size=32,
                                       color_mode='grayscale',
                                       class_mode='binary')
test_data = datagen.flow_from_directory('test',
                                        target_size=(150, 150),
                                        batch_size=32,
                                        color_mode='grayscale',
                                        class_mode='binary')

learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss',
                                            factor=0.5,
                                            patience=2,
                                            )
```

```
Found 5216 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

color_mode 옵션을 이용하면 굳이 배열을 통한 reshape 과정 없이 grayscale 이미지를 읽어들일 수 있다.

■ Model

- 세 가지 모델을 만들어 본 후 accuracy가 가장 좋은 모델을 추가 개선
- 시간 관계 상 epoch은 작게 4로 테스트

- Model 1

```
def model1():
  model = Sequential()

  model.add(Conv2D(32, kernel_size=(3, 3),
                   input_shape=(150, 150, 1),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(64, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(128, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dropout(rate=0.5))
  model.add(Dense(32, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))

  model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

  model.summary()

  return model
```

- Model 2

```
def model2():
  model = Sequential()

  model.add(Conv2D(32, kernel_size=(3, 3),
                input_shape=(150, 150, 1),
                activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(64, kernel_size=(3, 3),
                activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(128, kernel_size=(3, 3),
                activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(256, kernel_size=(3, 3),
                activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dropout(rate=0.5))
  model.add(Dense(32, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))

  model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

  model.summary()

  return model
```

- Model 3

```
def model3():
  model = Sequential()

  model.add(Conv2D(32, kernel_size=(3, 3),
                   input_shape=(150, 150, 1),
                   activation='relu'))
  model.add(BatchNormalization())
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(16, kernel_size=(3, 3),
                   activation='relu'))
  model.add(Dropout(rate=0.2))
  model.add(BatchNormalization())
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dense(64, activation='relu'))
  model.add(Dropout(rate=0.2))
  model.add(Dense(1, activation='sigmoid'))

  model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

  model.summary()

  return model
```

- Model summary

  - Model 1

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 36992)             0
_________________________________________________________________
dropout (Dropout)            (None, 36992)             0
_________________________________________________________________
dense (Dense)                (None, 32)                1183776
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,276,481
Trainable params: 1,276,481
Non-trainable params: 0
```

  - Model 2

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 148, 148, 32)      320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 15, 15, 256)       295168
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12544)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12544)             0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                401440
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 789,313
Trainable params: 789,313
Non-trainable params: 0
```

- Model 3

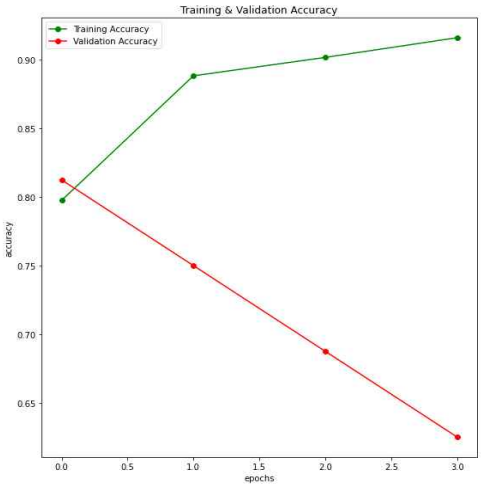
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 148, 148, 32)      320
_________________________________________________________________
batch_normalization_4 (Batch (None, 148, 148, 32)      128
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 72, 72, 16)        4624
_________________________________________________________________
dropout_6 (Dropout)          (None, 72, 72, 16)        0
_________________________________________________________________
batch_normalization_5 (Batch (None, 72, 72, 16)        64
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 36, 36, 16)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_8 (Dense)              (None, 64)                1327168
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 65
=================================================================
Total params: 1,332,369
Trainable params: 1,332,273
Non-trainable params: 96
```

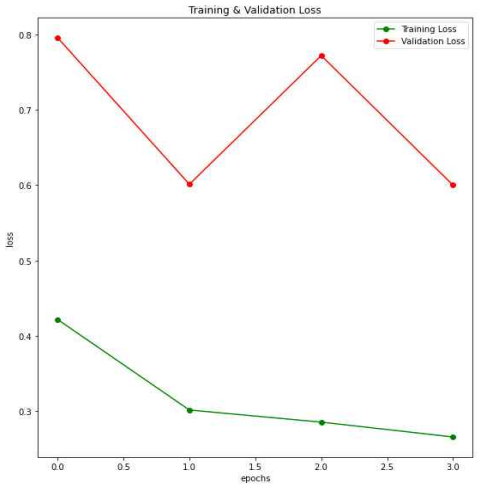
■ Learning

```
first_model = model1()
second_model = model2()
third_model = model3()

history1 = first_model.fit(train_data,
                        validation_data=val_data,
                        epochs=4,
                        batch_size=32,
                        callbacks=[learning_rate_reduction]
                        )

history2 = second_model.fit(train_data,
                        validation_data=val_data,
                        epochs=4,
                        batch_size=32,
                        callbacks=[learning_rate_reduction]
                        )

history3 = third_model.fit(train_data,
                        validation_data=val_data,
                        epochs=4,
                        batch_size=32,
                        callbacks=[learning_rate_reduction]
                        )
```

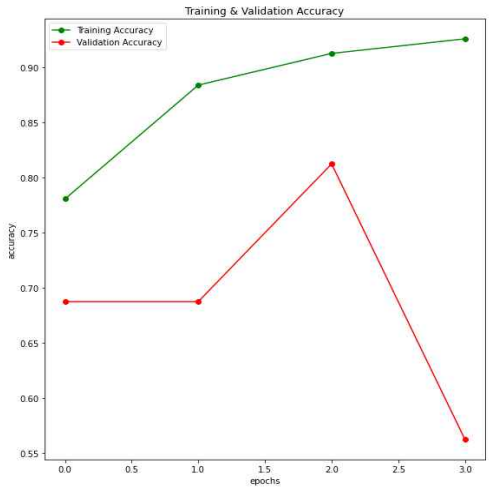
■ Test

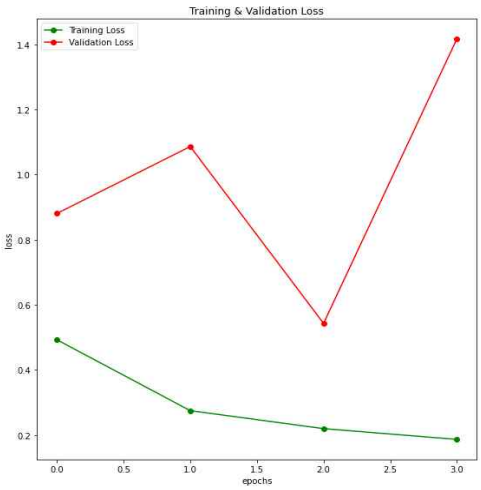
- Model 1



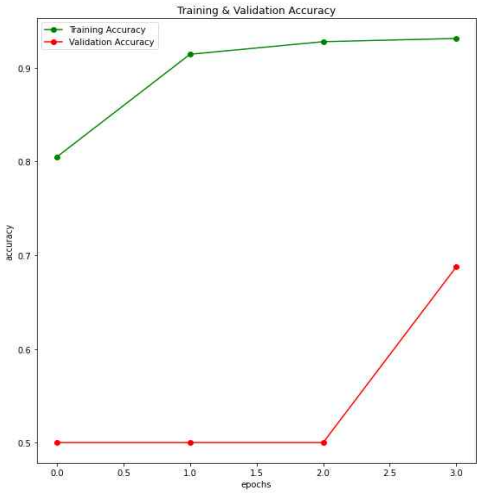


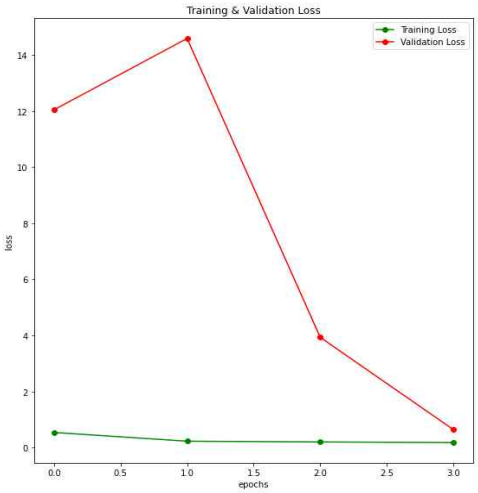
- Model 2





- Model 3

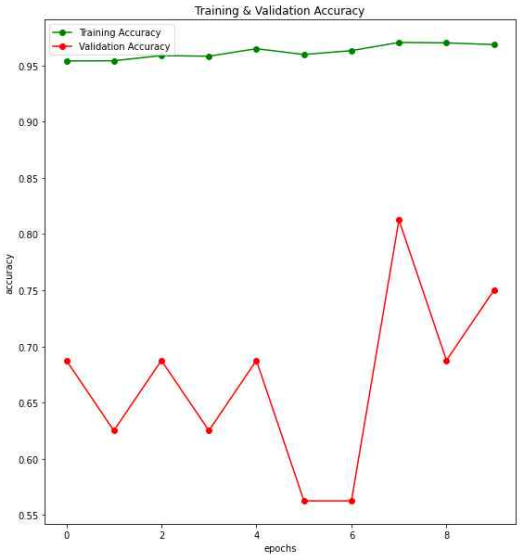


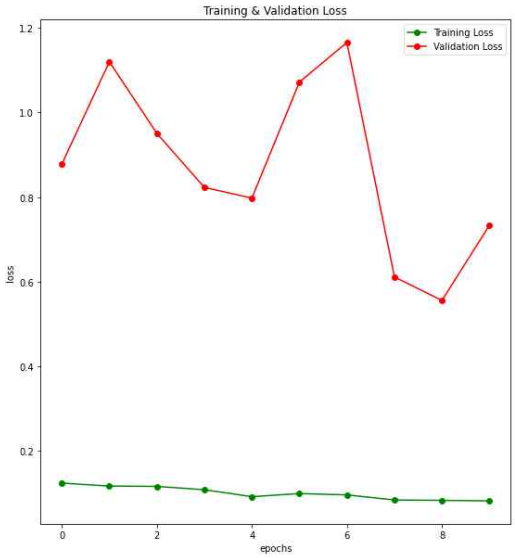


Model 1 같은 경우에는 validation accuracy가 현저히 감소하는 것을 볼 수 있다. Model 3의 loss가 꾸준히 감소하여 epoch을 늘려 테스트해봤는데 accuracy가 더 이상 높아지지 않았다. 그래서 Model 2를 개선시키기로 했다.

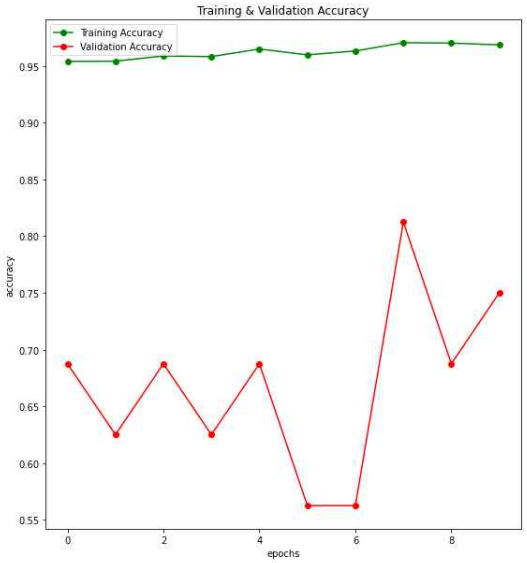
■ Re-test

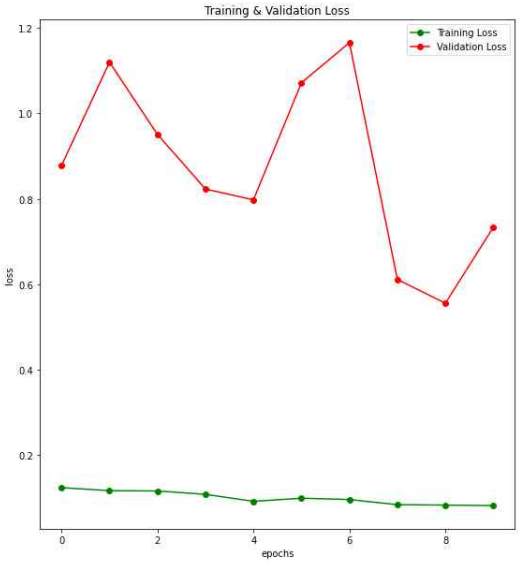
- epoch = 10





- epoch = 20






```
Epoch 13/20
163/163 [==============================] - 58s 356ms/step - loss: 0.0663 - accuracy: 0.9758 - val_loss: 0.7348 - val_accuracy: 0.7500 - lr: 1.5625e-05
Epoch 14/20
163/163 [==============================] - 58s 357ms/step - loss: 0.0627 - accuracy: 0.9778 - val_loss: 0.6257 - val_accuracy: 0.8125 - lr: 7.8125e-06
Epoch 15/20
163/163 [==============================] - 59s 361ms/step - loss: 0.0710 - accuracy: 0.9745 - val_loss: 0.5651 - val_accuracy: 0.8125 - lr: 7.8125e-06
Epoch 16/20
163/163 [==============================] - 59s 363ms/step - loss: 0.0659 - accuracy: 0.9728 - val_loss: 1.0727 - val_accuracy: 0.5625 - lr: 3.9063e-06
Epoch 17/20
163/163 [==============================] - 58s 357ms/step - loss: 0.0690 - accuracy: 0.9755 - val_loss: 0.5438 - val_accuracy: 0.7500 - lr: 3.9063e-06
Epoch 18/20
163/163 [==============================] - 58s 353ms/step - loss: 0.0662 - accuracy: 0.9753 - val_loss: 0.5894 - val_accuracy: 0.5625 - lr: 1.9531e-06
Epoch 19/20
163/163 [==============================] - 58s 354ms/step - loss: 0.0647 - accuracy: 0.9766 - val_loss: 0.9921 - val_accuracy: 0.6250 - lr: 1.9531e-06
Epoch 20/20
163/163 [==============================] - 58s 353ms/step - loss: 0.0688 - accuracy: 0.9743 - val_loss: 1.2711 - val_accuracy: 0.5625 - lr: 9.7656e-07
```

epoch을 키우니 확실히 validation accuracy가 증가하는 것을 볼 수 있다. epoch 15인 지점이 최고점이라고 판단하고, 이제 다른 매개변수를 바꿔봐야겠다.

■ Callback function

- ReduceLROnPlateau

- 학습률이 개선되지 않을 때 콜백함수를 이용하면 학습률을 동적으로 조정하여 개선할 수 있다. local minimum에 갇혀버리면 쉽게 빠져나오지 못하고 갇히기 때문에 learning rate를 조정하여 효과를 볼 수 있다고 한다.

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss',
                                            factor=0.5,
                                            patience=2,
                                            )
```

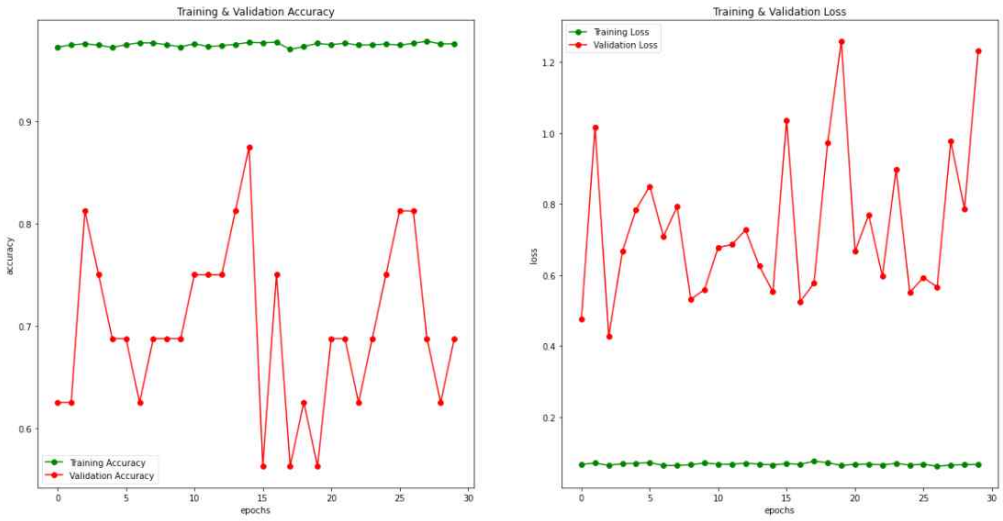
앞서 콜백을 위와 같이 정의했다.

- factor : 콜백 호출 시 학습률을 몇으로 줄일건지 정의한다.
- patience : 콜백이 호출될 epoch을 지정한다.

처음 모델을 설계했을 때 epoch 값을 4로 두었기 때문에 patience를 2로 했었다. 이제는 값을 5로 바꾸어 epoch 5 동안 성능 개선이 이루어지지 않으면 콜백 함수를 호출해보기로 했다.

```
Epoch 10/15
163/163 [==============================] - 61s 377ms/step - loss: 0.0689 - accuracy: 0.9753 - val_loss: 0.5645 - val_accuracy: 0.6875 - lr: 4.8828e-07
Epoch 11/15
163/163 [==============================] - 62s 379ms/step - loss: 0.0672 - accuracy: 0.9760 - val_loss: 0.6811 - val_accuracy: 0.7500 - lr: 4.8828e-07
Epoch 12/15
163/163 [==============================] - 62s 380ms/step - loss: 0.0734 - accuracy: 0.9722 - val_loss: 0.6869 - val_accuracy: 0.7500 - lr: 4.8828e-07
Epoch 13/15
163/163 [==============================] - 62s 379ms/step - loss: 0.0736 - accuracy: 0.9714 - val_loss: 0.7246 - val_accuracy: 0.7500 - lr: 4.8828e-07
Epoch 14/15
163/163 [==============================] - 62s 380ms/step - loss: 0.0699 - accuracy: 0.9726 - val_loss: 0.6242 - val_accuracy: 0.8125 - lr: 2.4414e-07
Epoch 15/15
163/163 [==============================] - 62s 380ms/step - loss: 0.0670 - accuracy: 0.9778 - val_loss: 0.5508 - val_accuracy: 0.8750 - lr: 2.4414e-07
```

마지막에 정확도가 급격히 증가한 것을 보고 넉넉히 epoch을 30으로 잡고 다시 측정했지만 예상과는 달리 더 이상 오르지 않았다.



■ Re-model

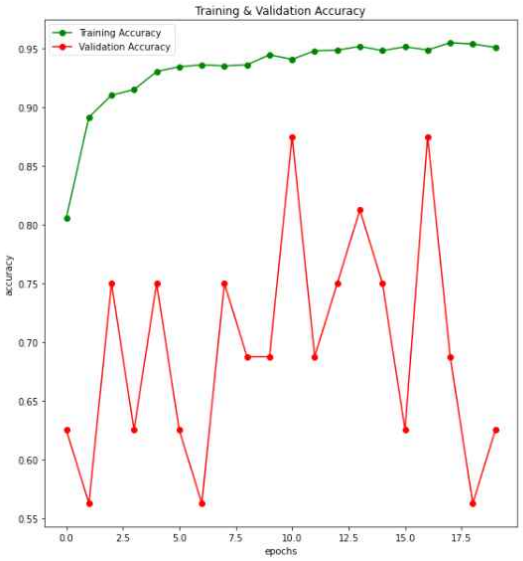
- Convolution layer 추가

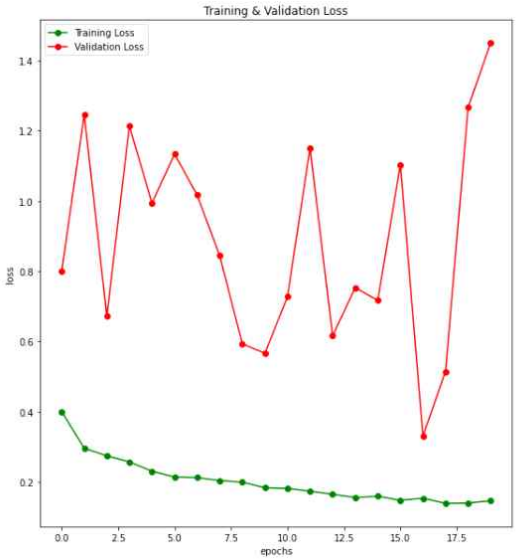
```
def model2_implement():
  model = Sequential()

  model.add(Conv2D(32, kernel_size=(3, 3),
                   input_shape=(150, 150, 1),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(64, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(128, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(256, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(512, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dropout(rate=0.5))
  model.add(Dense(256, activation='relu'))
  model.add(Dense(64, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))

  model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

  model.summary()

  return model
```

혹시나 하는 마음에 convolution layer를 추가, 제거 해봤는데 정확도는 비슷했고, loss만 더욱 증가했다.

- Convolution layer 제거

```
def model2_implement2():
  model = Sequential()

  model.add(Conv2D(32, kernel_size=(3, 3),
                   input_shape=(150, 150, 1),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(64, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dropout(rate=0.5))
  model.add(Dense(16, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))

  model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

  model.summary()

  return model
```





■ Conclusion code

- Design model

```
np.random.seed(100)

def final_model():
  model = Sequential()

  model.add(Conv2D(32, kernel_size=(3, 3),
                   input_shape=(150, 150, 1),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(64, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(128, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(256, kernel_size=(3, 3),
                   activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dropout(rate=0.5))
  model.add(Dense(32, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))

  model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

  model.summary()

  return model
```

- Prepare data

```
datagen = ImageDataGenerator(rescale=1./255,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True)
train_data = datagen.flow_from_directory('train',
                                         target_size=(150, 150),
                                         batch_size=32,
                                         color_mode='grayscale',
                                         class_mode='binary')
val_data = datagen.flow_from_directory('val',
                                       target_size=(150, 150),
                                       batch_size=32,
                                       color_mode='grayscale',
                                       class_mode='binary')
test_data = datagen.flow_from_directory('test',
                                        target_size=(150, 150),
                                        batch_size=32,
                                        color_mode='grayscale',
                                        class_mode='binary')

learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss',
                                            factor=0.5,
                                            patience=10,
                                            )
```

- Train model

```
model = final_model()

history = model.fit(train_data,
                    validation_data=val_data,
                    epochs=15,
                    batch_size=32,
                    callbacks=[learning_rate_reduction]
                    )
```

- Neural network architecture

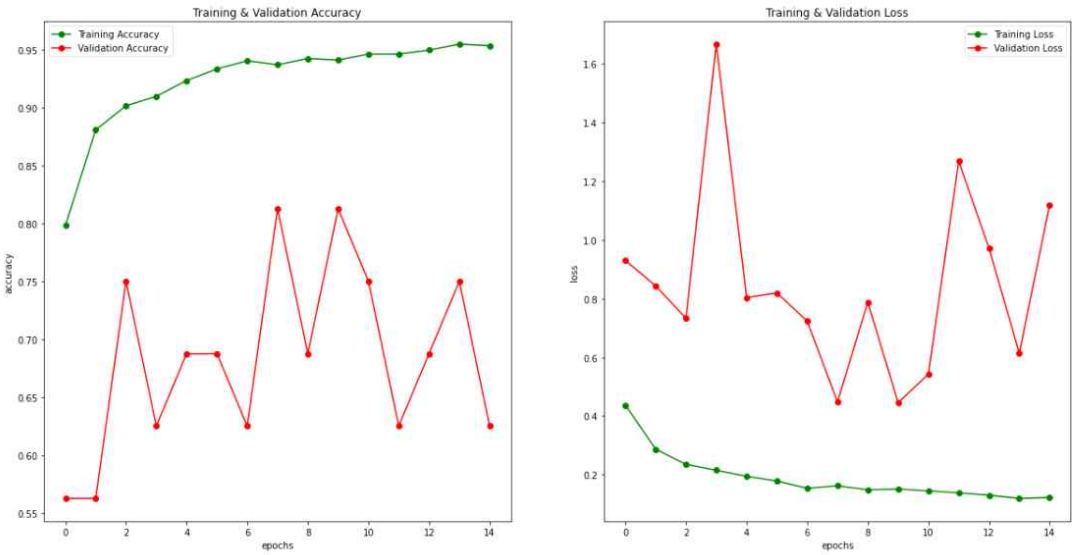
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 12544)             0
_________________________________________________________________
dropout (Dropout)            (None, 12544)             0
_________________________________________________________________
dense (Dense)                (None, 32)                401440
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 789,313
Trainable params: 789,313
Non-trainable params: 0
```

- Accuracy & Loss

```
Epoch 10/15
163/163 [==============================] - 63s 387ms/step - loss: 0.1512 - accuracy: 0.9410 - val_loss: 0.4452 - val_accuracy: 0.8125 - lr: 0.0010
Epoch 11/15
163/163 [==============================] - 63s 385ms/step - loss: 0.1451 - accuracy: 0.9461 - val_loss: 0.5427 - val_accuracy: 0.7500 - lr: 0.0010
Epoch 12/15
163/163 [==============================] - 62s 383ms/step - loss: 0.1383 - accuracy: 0.9461 - val_loss: 1.2677 - val_accuracy: 0.6250 - lr: 0.0010
Epoch 13/15
163/163 [==============================] - 62s 382ms/step - loss: 0.1305 - accuracy: 0.9496 - val_loss: 0.9715 - val_accuracy: 0.6875 - lr: 0.0010
Epoch 14/15
163/163 [==============================] - 62s 383ms/step - loss: 0.1195 - accuracy: 0.9548 - val_loss: 0.6141 - val_accuracy: 0.7500 - lr: 0.0010
Epoch 15/15
163/163 [==============================] - 63s 385ms/step - loss: 0.1226 - accuracy: 0.9534 - val_loss: 1.1191 - val_accuracy: 0.6250 - lr: 0.0010
```

- Graph

```
epochs = [x for x in range(15)]

fig, ax = plt.subplots(1, 2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20, 10)

ax[0].plot(epochs, train_acc, 'go-', label='Training Accuracy')
ax[0].plot(epochs, val_acc, 'ro-', label='Validation Accuracy')
ax[0].set_title('Training & Validation Accuracy')
ax[0].legend()
ax[0].set_xlabel('epochs')
ax[0].set_ylabel('accuracy')

ax[1].plot(epochs, train_loss, 'g-o', label='Training Loss')
ax[1].plot(epochs, val_loss, 'r-o', label='Validation Loss')
ax[1].set_title('Training & Validation Loss')
ax[1].legend()
ax[1].set_xlabel('epochs')
ax[1].set_ylabel('loss')
plt.show()

model.save('final-model.h5')
```



- Predict

```
evaluation = model.evaluate(train_data)
print('Train Accuracy: %.2f' % (evaluation[1] * 100))

evaluation = model.evaluate(test_data)
print('Test Accuracy: %.2f' % (evaluation[1] * 100))
```

```
163/163 [==============================] - 60s 370ms/step - loss: 0.1086 - accuracy: 0.9584
Train Accuracy: 95.84
20/20 [==============================] - 6s 299ms/step - loss: 0.5209 - accuracy: 0.8253
Test Accuracy: 82.53
```

■ Impression

- 중간고사 때 진행했던 머신러닝 경진대회처럼 이번에도 정확도 올리는 작업이 쉽지 않았다. 그래도 이번 학기에 영상정보처리 수업을 함께 들은 덕분에 이미지 데이터를 시각적으로 표현할 수 있는 점은 꽤 뿌듯했다.

수업 때 경험해본 실습 데이터들과는 다르게 이미지 데이터가 통으로 들어 있어 어떻게 처리해줘야 할지 고민했다. openCV를 함께 이용해 numpy 배열에 다 넣어서 작업하려고 했는데 생각보다 코드로 구현하기가 막막했다. 인터넷을 찾아본 결과 ImageDataGenerator를 클래스를 통해 데이터 전처리를 쉽게 할 수 있었고, 그 중 flow_flow_directory 메소드를 사용하면 폴더 형태로된 데이터를 바로 가져와서 사용할 수 있어 편리했다. 캐글 같은 플랫폼은 데이터를 폴더 구조로 올려주기 때문에 효과적이라고 생각한다.

처음에는 color_mode 값을 지정해주지 않아 정확도가 굉장히 낮았다. 왜냐하면 neural network 설계할 때 input_shape을 단일 채널로 지정해줬는데, RGB 3채널로 받아왔기 때문이다. 이 부분에서 시간 낭비한 것이 조금은 아쉽다.

그래도 한 학기 동안 머신러닝부터 딥러닝까지 인공지능에 대해 깊게 공부해 볼 수 있었다. 클라우드 관련 내용까지 진도를 나갔다면 어땠을까도 싶지만 학부 수준에서 이 정도 학습한 것도 좋은 경험이라고 생각한다.