In [1]:

```
######### HW05_곽용하_Kwak Yongha_2014121047 #########

###(a)###
## tf.keras.preprocessing.image.ImageDataGenerator ##
#the role: 이미지를 불러와 적절한 텐서의 배치로 변환하는 역할을 합니다.이미지데이터
#arguments
1) featurewise_center: input data의 평균을 0으로 함
2) brightness_range: 이미지 밝기 조절
3) width_shift_range: 임의의 크기만큼 너비 방향으로 이동
4) rescale: 픽셀 크기를 조정
5) preprocessing_function: 이미지 처리가 진행된 후, 각 input에 적용할 함수

###(b)###
[3] 리스케일링 하는 단계. validation을 위해 전체에서 20%를 할당합니다.
[4] train data set을 만드는 단계. 각 iteration마다 주는 데이터사이즈(batch size)는
[5] test data set을 만드는 단계. 위와 같은 방법으로 변형시켜 valid data로 저장합니

###(c)###
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D,MaxPool2D,Dropout,Flatten,Dense,BatchN
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Dividing Dataset into two folders train and test
width = 128
height = 128
datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.2)

trainDatagen = datagen.flow_from_directory(directory='C:/Users/david/Downloads/cel
                                           target_size=(width,height),
                                           class_mode = 'binary',
                                           batch_size = 16,
                                           subset='training')

valDatagen = datagen.flow_from_directory(directory='C:/Users/david/Downloads/cell_
                                           target_size=(width,height),
                                           class_mode = 'binary',
                                           batch_size = 16,
                                           subset='validation')
```

```
Found 22048 images belonging to 2 classes.
Found 5510 images belonging to 2 classes.
```

In [2]:
```
# define the keras model
model = Sequential([
    Conv2D(16, (3, 3), activation='relu',padding='same'),
    MaxPool2D(pool_size=(2, 2),strides=2,padding='same'),
    Conv2D(32, (3, 3), activation='relu',padding='same'),
    MaxPool2D(pool_size=(2, 2),strides=2,padding='same'),
    Conv2D(64, (3, 3), activation='relu',padding='same'),
    MaxPool2D(pool_size=(2, 2),strides=2,padding='same'),
    Conv2D(64, (3, 3), activation='relu',padding='same'),
    MaxPool2D(pool_size=(2, 2),strides=2,padding='same'),
    Flatten(),
    Dense(1024, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

In [6]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, None, None, 16)    448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, None, None, 16)    0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, None, None, 32)    4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, None, None, 32)    0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, None, None, 64)    18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, None, None, 64)    0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, None, None, 64)    36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, None, None, 64)    0
_________________________________________________________________
flatten (Flatten)            (None, None)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              4195328
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 1025
=================================================================
Total params: 4,256,865
Trainable params: 4,256,865
Non-trainable params: 0
_________________________________________________________________
```

In [3]:
```
###(d)###
#model compile & fit
model.compile(optimizer='adam',
              loss = 'binary_crossentropy',
              metrics=['accuracy'])

history = model.fit(trainDatagen,
steps_per_epoch=30,
epochs=20,
verbose=1,
validation_data = valDatagen,
validation_steps=30)
```

```
Epoch 1/20
30/30 [==============================] - 12s 325ms/step - loss: 0.7001 - accu
racy: 0.5354 - val_loss: 0.6940 - val_accuracy: 0.4938
Epoch 2/20
30/30 [==============================] - 9s 307ms/step - loss: 0.6741 - accur
acy: 0.5750 - val_loss: 0.6702 - val_accuracy: 0.5979
Epoch 3/20
30/30 [==============================] - 9s 304ms/step - loss: 0.6634 - accur
acy: 0.6000 - val_loss: 0.6629 - val_accuracy: 0.6417
Epoch 4/20
30/30 [==============================] - 9s 301ms/step - loss: 0.6402 - accur
acy: 0.6479 - val_loss: 0.6690 - val_accuracy: 0.5188
Epoch 5/20
30/30 [==============================] - 9s 309ms/step - loss: 0.6496 - accur
acy: 0.6250 - val_loss: 0.6662 - val_accuracy: 0.5854
Epoch 6/20
30/30 [==============================] - 9s 294ms/step - loss: 0.5730 - accur
acy: 0.7167 - val_loss: 0.6045 - val_accuracy: 0.6271
Epoch 7/20
30/30 [==============================] - 9s 303ms/step - loss: 0.4776 - accur
acy: 0.7667 - val_loss: 0.4380 - val_accuracy: 0.8625
Epoch 8/20
30/30 [==============================] - 9s 310ms/step - loss: 0.4066 - accur
acy: 0.8417 - val_loss: 0.2950 - val_accuracy: 0.8792
Epoch 9/20
30/30 [==============================] - 9s 306ms/step - loss: 0.2653 - accur
acy: 0.8854 - val_loss: 0.2539 - val_accuracy: 0.8979
Epoch 10/20
30/30 [==============================] - 9s 308ms/step - loss: 0.1730 - accur
acy: 0.9292 - val_loss: 0.2105 - val_accuracy: 0.9167
Epoch 11/20
30/30 [==============================] - 9s 308ms/step - loss: 0.2523 - accur
acy: 0.9125 - val_loss: 0.4833 - val_accuracy: 0.9375
Epoch 12/20
30/30 [==============================] - 9s 299ms/step - loss: 0.1278 - accur
acy: 0.9583 - val_loss: 0.2700 - val_accuracy: 0.9333
Epoch 13/20
30/30 [==============================] - 10s 326ms/step - loss: 0.1185 - accu
racy: 0.9563 - val_loss: 0.2089 - val_accuracy: 0.9271
Epoch 14/20
30/30 [==============================] - 10s 316ms/step - loss: 0.1995 - accu
racy: 0.9542 - val_loss: 0.1942 - val_accuracy: 0.9208
```

```
Epoch 15/20
30/30 [==============================] - 9s 301ms/step - loss: 0.1665 - accur
acy: 0.9458 - val_loss: 0.1847 - val_accuracy: 0.9417
Epoch 16/20
30/30 [==============================] - 9s 312ms/step - loss: 0.1842 - accur
acy: 0.9438 - val_loss: 0.1671 - val_accuracy: 0.9417
Epoch 17/20
30/30 [==============================] - 9s 297ms/step - loss: 0.1918 - accur
acy: 0.9438 - val_loss: 0.1794 - val_accuracy: 0.9417
Epoch 18/20
30/30 [==============================] - 9s 313ms/step - loss: 0.1388 - accur
acy: 0.9646 - val_loss: 0.1848 - val_accuracy: 0.9396
Epoch 19/20
30/30 [==============================] - 9s 313ms/step - loss: 0.1420 - accur
acy: 0.9479 - val_loss: 0.2229 - val_accuracy: 0.9229
Epoch 20/20
30/30 [==============================] - 9s 314ms/step - loss: 0.1342 - accur
acy: 0.9500 - val_loss: 0.2130 - val_accuracy: 0.9354
```

In [4]:

```
## report accuracy from training/test data
# training accuracy = 0.95, test accuracy = 0.9354


hist = history.history
loss = hist['loss']
val_loss = hist['val_loss']
epochs = range(1,len(loss)+1)
fig = plt.figure(figsize = (10,5))

graph1 = fig.add_subplot(1,2,1)
graph1.plot(epochs, loss, color='red', label='train loss')
graph1.plot(epochs, val_loss, color = 'blue', label='val_loss')
graph1.set_title('train_loss & val_loss')
graph1.set_xlabel('epochs')
graph1.set_ylabel('loss')
graph1.legend()

accu = hist['accuracy']
val_accu = hist['val_accuracy']

graph2 = fig.add_subplot(1,2,2)
graph2.plot(epochs, accu, color='red', label='train accuracy')
graph2.plot(epochs, val_accu, color = 'blue', label='val_accuracy')
graph2.set_title('train_accuracy & val_accuracy')
graph2.set_xlabel('epochs')
graph2.set_ylabel('accuracy')
graph2.legend()

plt.show()
```