x2=[]

for i in range(len(x)):

img=cv2.resize(x[i],(224,224))

x2.append(img)

train\_test\_splitx\_train,x\_test,y\_train,y\_test=train\_test\_split(x2,y2,test\_size=0.1,random\_state=1, stratify=y2)

**cv2.resize 를 통해 input size를 (224,224)로 변경하였다. 그 후, test set을 분리하였고, vgg16 이전학습을 거친 후, validation split을 통해 validation set을 분리한 상태로 다음 모델에 학습하였다.**

modelvgg16=Sequential()

modelvgg16.add(Conv2D(512,(3,3),activation='relu',input\_shape=(x\_train.shape[1],x\_train.shape[2],x\_train.shape[3])))

modelvgg16.add(MaxPooling2D((2,2)))

modelvgg16.add(Flatten())

modelvgg16.add(Dense(512,activation='relu')

modelvgg16.add(Dropout(0.5))

modelvgg16.add(Dense(1,activation='sigmoid'))

modelvgg16.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

history\_vgg16=modelvgg16.fit(x\_train/np.max(x\_train),y\_train,batch\_size=32,epochs=20,validation\_split=0.1)

**학습한 모델에 대하여 test set에 evaluate 시켰더니, train set의 acc는 1 인데 비해, test set의 acc는 0.94가 나왔고, overfitting이 발생하였음을 알 수 있다.**

modelvgg16.evaluate(x\_test/np.max(x\_train),y\_test,batch\_size=32)

/6 [==============================] - 0s 14ms/step - loss: 0.4296 - accuracy: 0.9364Out[12]:[0.4296308755874634, 0.9364162087440491]

**이제 잔차연결을 활용한 모델에 train data를 학습시키자. 모델의 중간중간에 overfitting을 막기 위해 dropout을 주었고, 추정하려는 y가 개나 고양이(1 또는 0)이기에 flatten을 시켜서 dense층을 만들었다. 따라서 activation은 sigmoid, loss=binary\_crossentropy를 주었다.**

.......

conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel\_initializer='he\_normal')(conv9)

dropout=Dropout(0.5)(conv9)

conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel\_initializer='he\_normal')(dropout)

conv9 = BatchNormalization()(conv9)  
flatten=Flatten()

conv9=flatten(conv9)

dropout=Dropout(0.5)(conv9)

dense2=Dense(1,activation='sigmoid')(dropout)

model = Model(inputs = base\_vgg16.input, outputs = dense2)

**그 다음 모델을 컴파일 시키고 validation\_split=0.1을 주고 학습을 하였다. 그 후 test set에 evaluation 을 시켰더니 test set의 acc가 0.8035가 나왔고, 전 모형보다 overfitting이 더 심하게 일어난 것을 알 수 있다. 데이터는 1700 여개인데에 비해 마지막 dense층의 모수 수가 심각하게 많아서 일어난 현상 같아 보인다. Dropout 을 0.5를 주었는데도 overfitting을 해결하지 못하였다.**

6/6 [==============================] - 10s 2s/step - loss: 14.0856 - accuracy: 0.8035

[14.085561752319336, 0.8034682273864746]

**Bydirectional rnn 모델에서 validation split을 통해 validation set을 분리하였고, merge\_mode를 바꿔가면서 결과를 확인해보았다. concat, sum, ave 등을 시도해보았지만, 모두 비슷한 evaluate 값을 내보냈다. L1 regularuzation을 적용해도 overfitting을 해결하지 못하였다.**

m\_bd.add(Bidirectional(LSTM(32),merge\_mode='concat'))

m\_bd.add(Dense(1, activation='sigmoid',kernel\_regularizer=l1(0.1)))

m\_bd.compile(optimizer='rmsprop',loss='binary\_crossentropy',metrics=['acc'])

m\_bd.fit(x\_train,y\_train, epochs=30,batch\_size=128,validation\_split=0.15)

196/196 [==============================] - 40s 206ms/step - loss: 0.3700 - acc: 0.8414

Out[7]:[0.370006799697876, 0.8414400219917297]

m\_bd2.add(Bidirectional(LSTM(32),merge\_mode='sum'))

196/196 [==============================] - 41s 207ms/step - loss: 0.5172 - acc: 0.8463

Out[9]:[0.5172477960586548, 0.8462799787521362]

…