**딥러닝/클라우드**

기말대체과제 레포트

32183164 이석현

Dankook University

2020 Fall

목차

1. **Neural network structure 2**

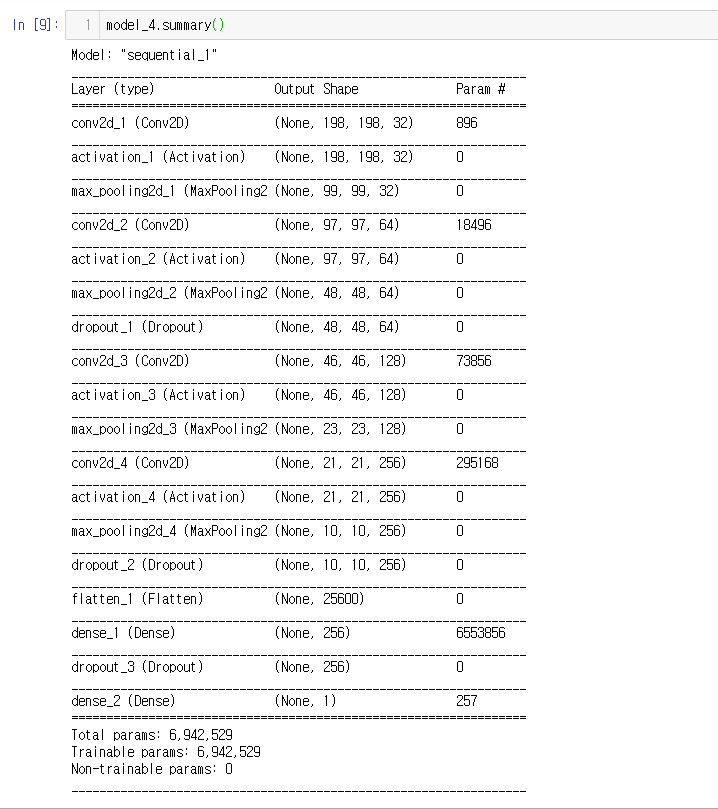
**2. Source code 3**

**3. Last 5 epochs, Test loss, Test accuracy 9**

**4. 학습곡선그래프 9**

**5. 소감 10**

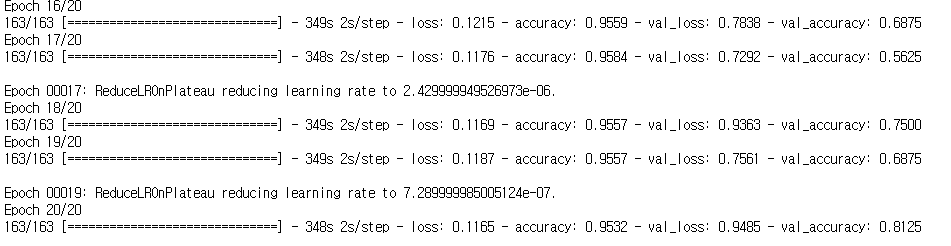
**1. Neural network structure**

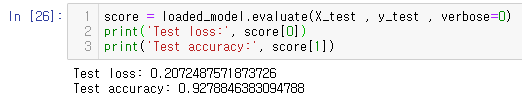


**2. Source code**

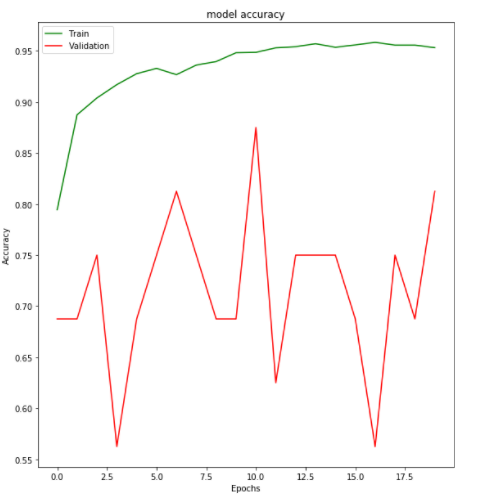
|  |
| --- |
| import numpy as np  import pandas as pd  import tensorflow as tf  from tensorflow import keras  from keras.models import Sequential  from keras.layers import Dense,Activation,Flatten,Dropout  from keras.layers import Conv2D,MaxPooling2D  from keras.callbacks import ModelCheckpoint  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras import layers  import os  from keras.utils import np\_utils  import os # operating system interfaces  # Set data folder  img\_dir\_train\_n = 'D:\\data\\chest\_xray\\train\\NORMAL'  img\_dir\_val\_n = 'D:\\data\\chest\_xray\\val\\NORMAL'  img\_dir\_test\_n = 'D:\\data\\chest\_xray\\test\\NORMAL'  img\_dir\_train\_p = 'D:\\data\\chest\_xray\\train\\PNEUMONIA'  img\_dir\_val\_p = 'D:\\data\\chest\_xray\\val\\PNEUMONIA'  img\_dir\_test\_p = 'D:\\data\\chest\_xray\\test\\PNEUMONIA'  # get file names  flist\_train\_n = os.listdir(img\_dir\_train\_n)  flist\_val\_n = os.listdir(img\_dir\_val\_n)  flist\_test\_n = os.listdir(img\_dir\_test\_n)  flist\_train\_p = os.listdir(img\_dir\_train\_p)  flist\_val\_p = os.listdir(img\_dir\_val\_p)  flist\_test\_p = os.listdir(img\_dir\_test\_p)  from keras.preprocessing import image  X\_train = np.zeros(shape=((len(flist\_train\_p)+len(flist\_train\_n),200,200,3)))  y\_train = np.zeros(shape=(len(flist\_train\_p)+len(flist\_train\_n)))  for idx , fname in enumerate(flist\_train\_n):  img\_path = os.path.join(img\_dir\_train\_n , fname)  img = image.load\_img(img\_path , target\_size=(200,200))  img\_array\_train = image.img\_to\_array(img)  img\_array\_train = np.expand\_dims(img\_array\_train , axis=0)  X\_train[idx] = img\_array\_train  y\_train[idx] = 0  for idx , fname in enumerate(flist\_train\_p):  img\_path = os.path.join(img\_dir\_train\_p , fname)  img = image.load\_img(img\_path , target\_size=(200,200))  img\_array\_train = image.img\_to\_array(img)  img\_array\_train = np.expand\_dims(img\_array\_train , axis=0)  X\_train[idx+len(flist\_test\_n)] = img\_array\_train  y\_train[idx+len(flist\_test\_n)] = 1  X\_train = X\_train / 255.0  from keras.preprocessing import image  X\_val = np.zeros(shape=((len(flist\_val\_p)+len(flist\_val\_n),200,200,3)))  y\_val = np.zeros(shape=(len(flist\_val\_p)+len(flist\_val\_n)))  for idx , fname in enumerate(flist\_val\_n):  img\_path = os.path.join(img\_dir\_val\_n , fname)  img = image.load\_img(img\_path , target\_size=(200,200))  img\_array\_val = image.img\_to\_array(img)  img\_array\_val = np.expand\_dims(img\_array\_val , axis=0)  X\_val[idx] = img\_array\_val  y\_val[idx] = 0  for idx , fname in enumerate(flist\_val\_p):  img\_path = os.path.join(img\_dir\_val\_p , fname)  img = image.load\_img(img\_path , target\_size=(200,200))  img\_array\_val = image.img\_to\_array(img)  img\_array\_val = np.expand\_dims(img\_array\_val , axis=0)  X\_val[idx+len(flist\_val\_n)] = img\_array\_val  y\_val[idx+len(flist\_val\_n)] = 1  X\_val = X\_val / 255.0  from keras.preprocessing import image  X\_test = np.zeros(shape=((len(flist\_test\_p)+len(flist\_test\_n),200,200,3)))  y\_test = np.zeros(shape=(len(flist\_test\_p)+len(flist\_test\_n)))  for idx , fname in enumerate(flist\_test\_n):  img\_path = os.path.join(img\_dir\_test\_n , fname)  img = image.load\_img(img\_path , target\_size=(200,200))  img\_array\_test = image.img\_to\_array(img)  img\_array\_test = np.expand\_dims(img\_array\_test , axis=0)  X\_test[idx] = img\_array\_test  y\_test[idx] = 0  for idx , fname in enumerate(flist\_test\_p):  img\_path = os.path.join(img\_dir\_test\_p , fname)  img = image.load\_img(img\_path , target\_size=(200,200))  img\_array\_test = image.img\_to\_array(img)  img\_array\_test = np.expand\_dims(img\_array\_test , axis=0)  X\_test[idx+len(flist\_test\_n)] = img\_array\_test  y\_test[idx+len(flist\_test\_n)] = 1  X\_test = X\_test / 255.0  train\_generator = ImageDataGenerator(  rotation\_range=2,  horizontal\_flip=True,  zoom\_range=.1 )  val\_generator = ImageDataGenerator(  rotation\_range=2,  horizontal\_flip=True,  zoom\_range=.1)  test\_generator = ImageDataGenerator(  rotation\_range=2,  horizontal\_flip= True,  zoom\_range=.1)  train\_generator.fit(X\_train)  val\_generator.fit(X\_val)  test\_generator.fit(X\_test)  model\_4=Sequential()  #The first CNN layer  model\_4.add(Conv2D(32,(3,3),input\_shape=(200,200,3)))  model\_4.add(Activation('relu'))  model\_4.add(MaxPooling2D(pool\_size=(2,2)))  #The second convolution layer  model\_4.add(Conv2D(64,(3,3)))  model\_4.add(Activation('relu'))  model\_4.add(MaxPooling2D(pool\_size=(2,2)))  model\_4.add(Dropout(0.3))  #The Third convolution layer  model\_4.add(Conv2D(128,(3,3)))  model\_4.add(Activation('relu'))  model\_4.add(MaxPooling2D(pool\_size=(2,2)))  #The Fourth convolution layer  model\_4.add(Conv2D(256,(3,3)))  model\_4.add(Activation('relu'))  model\_4.add(MaxPooling2D(pool\_size=(2,2)))  model\_4.add(Dropout(0.3))  model\_4.add(Flatten())  model\_4.add(Dense(256,activation='relu'))  model\_4.add(Dropout(0.5))  model\_4.add(Dense(1,activation='sigmoid'))  model\_4.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])  epochs = 20  batch\_size = 32  disp = model\_4.fit(X\_train , y\_train ,batch\_size = batch\_size, epochs=epochs, verbose=1, validation\_data = (X\_val , y\_val))  score = model\_4.evaluate(X\_test , y\_test , verbose=0)  print('Test loss:', score[0])  print('Test accuracy:', score[1])  plt.plot(disp.histroy['accuracy'])  plt.plot(disp.history['val\_accuracy'])  plt.title('model accuracy')  plt.ylabel('accuracy')  plt.xlabel('epoch')  plt.legend(['train', 'validation'], loc='upper left')  plt.show()  score = model\_4.evaluate(X\_test , y\_test , verbose=0)  print('Test loss:', score[0])  print('Test accuracy:', score[1]) |
| #Test loss & test accuracy  evaluation = loaded\_model.evaluate(test\_generator)  print(f"Test Loss: {evaluation[0] \* 100:.2f}%")  evaluation = loaded\_model.evaluate(test\_generator)  print(f"Test Accuracy: {evaluation[1] \* 100:.2f}%") |
| #save model  model\_json = model\_4.to\_json()  with open("nineth\_model.json", "w") as json\_file :  json\_file.write(model\_json)  model\_4.save\_weights("nineth\_model.h5")  print("Saved model to disk") |
| #Load model  from keras.models import model\_from\_json  json\_file = open("nineth\_model.json", "r")  loaded\_model\_json = json\_file.read()  json\_file.close()  loaded\_model = model\_from\_json(loaded\_model\_json)  loaded\_model.load\_weights("nineth\_model.h5")  print("Loaded model from disk")  #compile model  loaded\_model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy']) |

**3. Last 5 epochs, Test loss, Test accuracy**



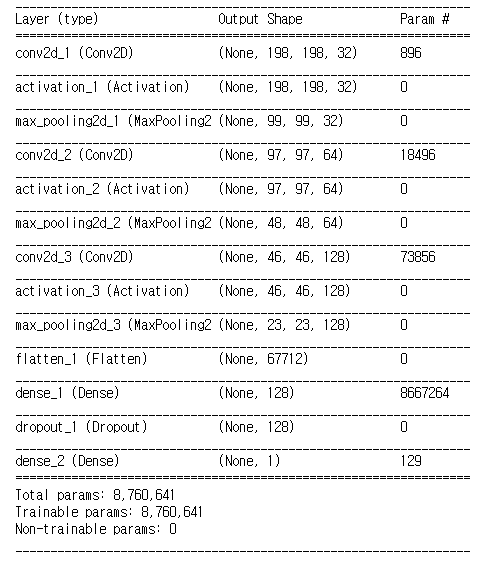
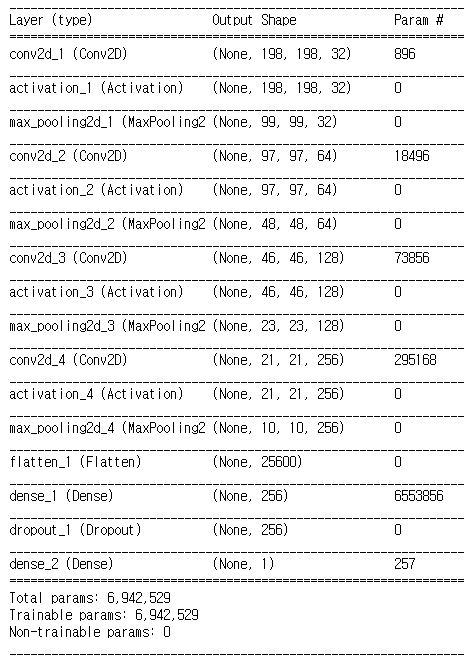
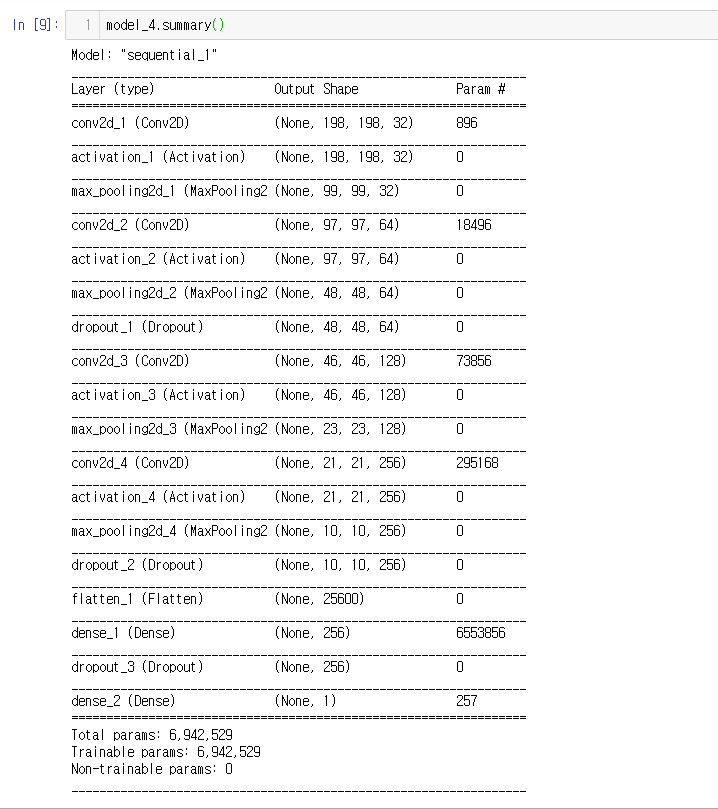


**4. 학습곡선그래프**



**5. 소감**

이번 프로젝트를 진행하면서 이미 training set, validation set, test set이 나누어져 있고 각 directory로부터 데이터를 generate할 때, seed값을 42로 고정해 주었기 때문에 매번 동일하게 accuracy가 형성될 것이라고 생각하였는데, 학습할 때 마다 accuracy의 추세가 달라져 학습을 시키는데 어려움이 있었습니다. 학습시킨 모델을 잃지 않기 위해서 중간중간 모델을 저장하다 보니 여덟 번째 모델까지 오게 되었습니다. 처음에는 아래의 첫번째 모델 구조로 학습을 했는데 validation accuracy가 처음부터 적게 나오고 epoch 수가 늘어나도 높아지지 않아서, convolution layer를 추가했습니다. 두번째 모델 구조로 학습을 하니 validation accuracy가 오르다 감소하는overfitting이 나타났고, 이를 해결하기위해 epoch를 낮추어 실행해보았더니 학습이 잘 되지 않아 train accuracy 자체가 생각만큼 높지 않게 형성되었습니다. 따라서 다시 epoch를 늘리고 overfitting이 되는 것을 줄이기 위해 convolution layer사이에 두 군데에 dropout을 넣어주었고 accuracy를 높일 수 있었습니다. 또 가장 높은 accuracy를 찾기 위해 epoch를 바꿔가며 여러 차례 모델링을 하였고 92%의 accuracy를 가진 모델을 개발하게 되었습니다. 처음부터 모델 structure 구상 및 accuracy를 높이기 위한 모델 구조 수정 작업이 시간도 오래 걸리고, 때로는 오히려 accuracy가 감소하는 경우도 있어 허무함이 들기도 하였습니다. 하지만 할 수 있다는 생각을 가지고 overfitting시 dropout을 늘리는 등 수업시간에 배웠던 내용을 토대로 수정을 하면서 accuracy를 높이겠다는 목표를 달성할 수 있었습니다. 마지막까지 같은 모델로 테스트를 하여도 accuracy가 조금씩 변한다는 문제가 있었는데, 테스트하는 부분에 문제가 있을 거라는 생각에 test\_generator부분의 shuffle=false로 설정하거나 batch\_size를 변경해보아도 해결할 수 없었습니다. 구글링을 했을 때 model structure에서 dropout을 제거할 경우 해결할 수 있다는 글을 보고 시도해 보았지만 그것 또한 실패했습니다. 그래서 다시 한번 고민을 하다가 파일을 로드하는 과정에서 문제가 생기지 않을까 라는 생각이 들었고 os.walk 대신 for문을 이용해 각 데이터를 로드하는 방식으로 바꾸면서 해결할 수 있었습니다. 이 과정에서 정말 많은 시간을 투자했고 포기할까 라는 생각도 하였지만 어떻게든 해결하겠다는 의지로 도전한 끝에 해낼 수 있었고 정말 뿌듯한 시간이었습니다.



△첫 번째 모델 구조

△두 번째 모델 구조

△세 번째 모델 구조

이번 학기가 시작할 때 딥러닝은 물론이고 머신러닝도 전혀 모르던 제 자신이 이 과목을 잘 수강할 수 있을지 걱정이 되었는데, 교수님께서 기초부터 차근차근 잘 설명해 주셔서 수업을 잘 따라갈 수 있었고 모델도 스스로 개발해보는 경험을 할 수 있었던 것 같습니다. 지금껏 전공 과목을 배워도 큰 관심이 없었는데 딥러닝에 흥미가 생겼고 더 깊게 공부해보고 싶다는 생각이 들어 이번 방학에는 kaggle에서 다른 데이터를 이용해 직접 모델을 개발하는 작업을 추가로 해보려고 합니다. 이번 학기 전면 온라인 강의임에도 교수님의 열정적인 가르침에 감사드립니다.