딥러닝/클라우드

기말대체과제 레포트

32183164 이석현

Dankook University

2020 Fall

딥러닝/클라우드 기말대체과제 레포트

Seokhyeon Lee(이석현) 32183164

목차

딥러닝/클라우드 기말대체과제 레포트

Seokhyeon Lee(이석현) 32183164

1. Neural network structure

```
In [9]: 1 model_4.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 198, 198, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 198, 198, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 99, 99, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 97, 97, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 97, 97, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 48, 48, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 46, 46, 128)       73856
_________________________________________________________________
activation_3 (Activation)    (None, 46, 46, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 23, 23, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 21, 21, 256)       295168
_________________________________________________________________
activation_4 (Activation)    (None, 21, 21, 256)       0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 10, 10, 256)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 10, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               6553856
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 6,942,529
Trainable params: 6,942,529
Non-trainable params: 0
_________________________________________________________________
```

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=['acc
uracy'])

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")
```

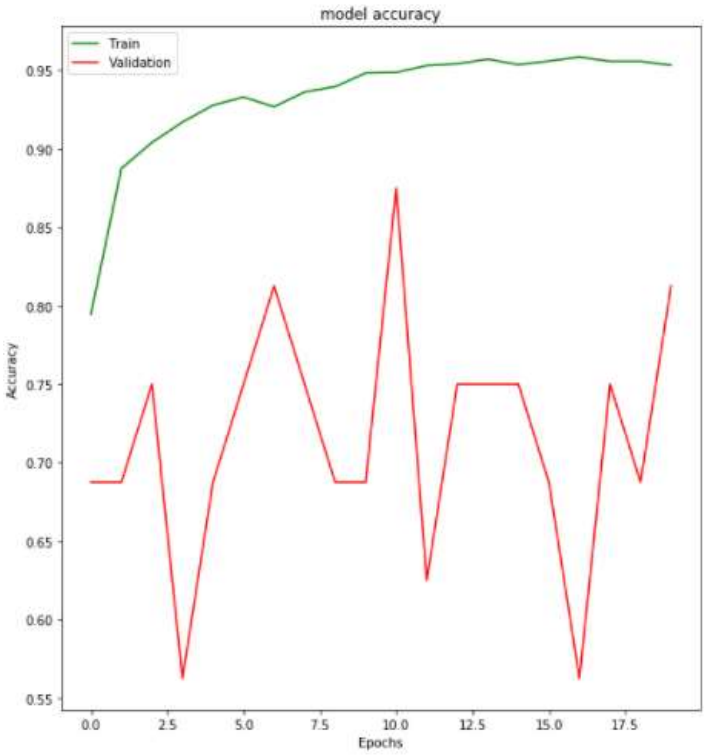
```
#Load model
from keras.models import model_from_json

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

```
Epoch 16/20
163/163 [==============================] - 349s 2s/step - loss: 0.1215 - accuracy: 0.9559 - val_loss: 0.7838 - val_accuracy: 0.6875
Epoch 17/20
163/163 [==============================] - 348s 2s/step - loss: 0.1176 - accuracy: 0.9584 - val_loss: 0.7292 - val_accuracy: 0.5625

Epoch 00017: ReduceLROnPlateau reducing learning rate to 2.429999949526973e-06.
Epoch 18/20
163/163 [==============================] - 349s 2s/step - loss: 0.1169 - accuracy: 0.9557 - val_loss: 0.9363 - val_accuracy: 0.7500
Epoch 19/20
163/163 [==============================] - 349s 2s/step - loss: 0.1187 - accuracy: 0.9557 - val_loss: 0.7561 - val_accuracy: 0.6875

Epoch 00019: ReduceLROnPlateau reducing learning rate to 7.289999985005124e-07.
Epoch 20/20
163/163 [==============================] - 348s 2s/step - loss: 0.1165 - accuracy: 0.9532 - val_loss: 0.9485 - val_accuracy: 0.8125
```

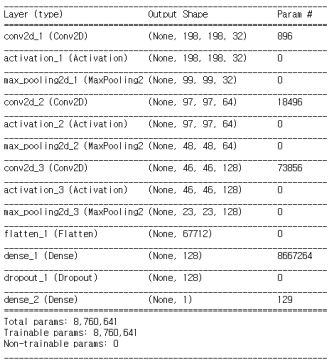
```
In [26]:
score = loaded_model.evaluate(X_test , y_test , verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

Test loss: 0.20724875718737266
Test accuracy: 0.9278846383094788
```

4. 학습곡선그래프



5. 소감

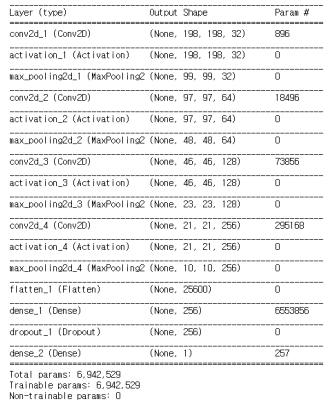
이번 프로젝트를 진행하면서 이미 training set, validation set, test set 이 나누어져 있고 각 directory 로부터 데이터를 generate 할 때, seed 값을 42 로 고정해 주었기 때문에 매번 동일하게 accuracy 가 형성될 것이라고 생각하였는데, 학습할 때 마다 accuracy 의 추세가 달라져 학습을 시키는데 어려움이 있었습니다. 학습시킨 모델을 잃지 않기 위해서 중간중간 모델을 저장하다 보니 여덟 번째 모델까지 오게 되었습니다. 처음에는 아래의 첫번째 모델 구조로 학습을 했는데 validation accuracy 가 처음부터 적게 나오고 epoch 수가 늘어나도 높아지지 않아서, convolution layer 를 추가했습니다. 두번째 모델 구조로 학습을 하니 validation accuracy 가 오르다 감소하는 overfitting 이 나타났고, 이를 해결하기위해 epoch 를 낮추어 실행해보았더니 학습이 잘 되지 않아 train accuracy 자체가 생각만큼 높지 않게 형성되었습니다. 따라서 다시 epoch 를 늘리고 overfitting 이 되는 것을 줄이기 위해 convolution layer 사이에 두 군데에 dropout 을 넣어주었고 accuracy 를 높일 수 있었습니다. 또 가장 높은 accuracy 를 찾기 위해 epoch 를 바꿔가며 여러 차례 모델링을 하였고 92%의 accuracy 를 가진 모델을 개발하게 되었습니다. 처음부터 모델 structure 구상 및 accuracy 를 높이기 위한 모델 구조 수정 작업이 시간도 오래 걸리고, 때로는 오히려 accuracy 가 감소하는 경우도 있어 허무함이 들기도 하였습니다. 하지만 할 수 있다는 생각을 가지고 overfitting 시 dropout 을 늘리는 등 수업시간에 배웠던 내용을 토대로 수정을 하면서 accuracy 를 높이겠다는 목표를 달성할 수 있었습니다. 마지막까지 같은 모델로 테스트를 하여도 accuracy 가 조금씩 변한다는 문제가 있었는데, 테스트하는 부분에 문제가 있을 거라는 생각에 test_generator 부분의 shuffle=false 로 설정하거나 batch_size 를 변경해보아도 해결할 수 없었습니다. 구글링을 했을 때 model structure 에서 dropout 을 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 for 문을 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.



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 198, 198, 32)	896
activation_1 (Activation)	(None, 198, 198, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 99, 99, 32)	0
conv2d_2 (Conv2D)	(None, 97, 97, 64)	18496
activation_2 (Activation)	(None, 97, 97, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 48, 48, 64)	0
conv2d_3 (Conv2D)	(None, 46, 46, 128)	73856
activation_3 (Activation)	(None, 46, 46, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 23, 23, 128)	0
flatten_1 (Flatten)	(None, 67712)	0
dense_1 (Dense)	(None, 128)	8667264
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129

Total params: 8,760,641
Trainable params: 8,760,641
Non-trainable params: 0

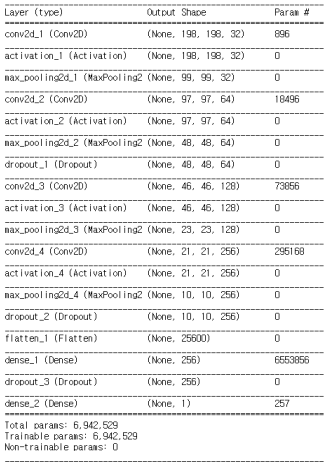
△첫 번째 모델 구조



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 198, 198, 32)	896
activation_1 (Activation)	(None, 198, 198, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 99, 99, 32)	0
conv2d_2 (Conv2D)	(None, 97, 97, 64)	18496
activation_2 (Activation)	(None, 97, 97, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 48, 48, 64)	0
conv2d_3 (Conv2D)	(None, 46, 46, 128)	73856
activation_3 (Activation)	(None, 46, 46, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 23, 23, 128)	0
conv2d_4 (Conv2D)	(None, 21, 21, 256)	295168
activation_4 (Activation)	(None, 21, 21, 256)	0
max_pooling2d_4 (MaxPooling2	(None, 10, 10, 256)	0
flatten_1 (Flatten)	(None, 25600)	0
dense_1 (Dense)	(None, 256)	6553856
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 1)	257

Total params: 6,942,529
Trainable params: 6,942,529
Non-trainable params: 0

△두 번째 모델 구조



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 198, 198, 32)	896
activation_1 (Activation)	(None, 198, 198, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 99, 99, 32)	0
conv2d_2 (Conv2D)	(None, 97, 97, 64)	18496
activation_2 (Activation)	(None, 97, 97, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 48, 48, 64)	0
dropout_1 (Dropout)	(None, 48, 48, 64)	0
conv2d_3 (Conv2D)	(None, 46, 46, 128)	73856
activation_3 (Activation)	(None, 46, 46, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 23, 23, 128)	0
conv2d_4 (Conv2D)	(None, 21, 21, 256)	295168
activation_4 (Activation)	(None, 21, 21, 256)	0
max_pooling2d_4 (MaxPooling2	(None, 10, 10, 256)	0
dropout_2 (Dropout)	(None, 10, 10, 256)	0
flatten_1 (Flatten)	(None, 25600)	0
dense_1 (Dense)	(None, 256)	6553856
dropout_3 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 1)	257

Total params: 6,942,529
Trainable params: 6,942,529
Non-trainable params: 0

△세 번째 모델 구조

딥러닝/클라우드 기말대체과제 레포트

Seokhyeon Lee(이석현) 32183164

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 kaggle 에서 다른 데이터를 이용해 직접 모델을 개발하는 작업을 추가로 해보려고 합니다. 이번 학기 전면 온라인 강의임에도 교수님의 열정적인 가르침에 감사드립니다.