13주차 과제

학번 : 2020254016
이름 : 박민우

1. 프로그램 6-1. LeNet-5로 MNIST 인식

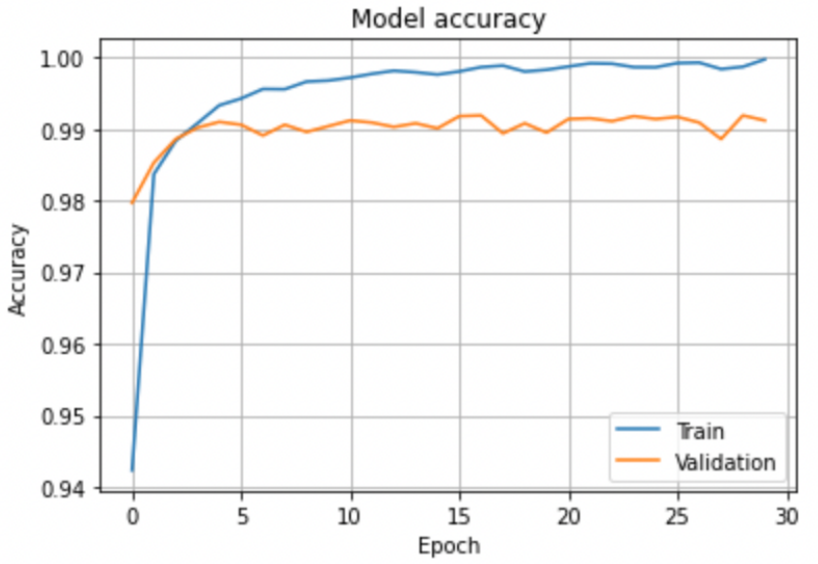
   1) 프로그램 동작 설명

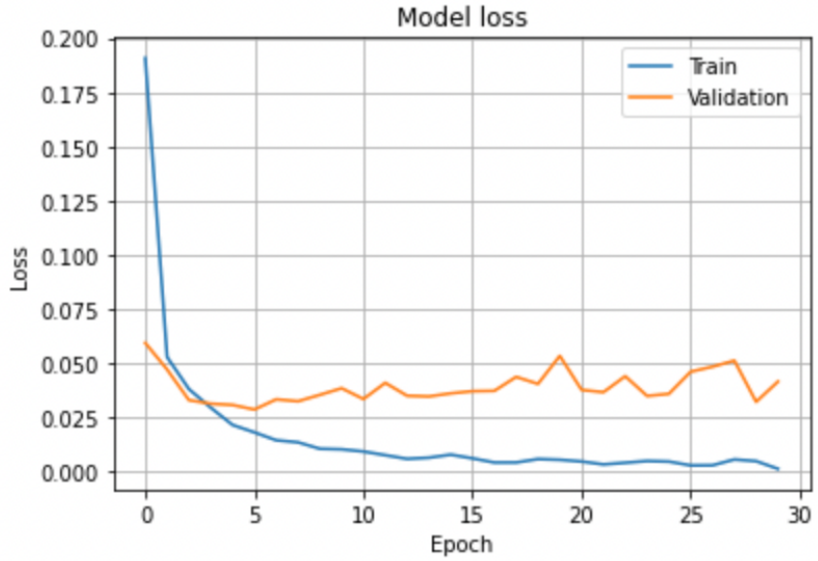
   - 프로그램 6-1은 MNIST를 인식하는 컨볼루션 신경망입니다.
   - 2차원 맵을 그대로 입력해야 하므로 10 ~ 11행에서 reshape(60000, 28, 28, 1)을 사용하였습니다.
   - 18 ~ 26행이 신경망 설계 부분으로 C-P-C-P-C-FC-FC 순서로 층을 쌓아 LeNet-5와 비슷한 구조로 설계되어 있습니다.

```
1 import numpy as np
2 import tensorflow as tf
3 from tensorflow.keras.datasets import mnist
4 from tensorflow.keras.models import Sequential
5 from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense
6 from tensorflow.keras.optimizers import Adam
7
8 # MNIST 데이터셋을 읽고 신경망에 입력할 형태로 변환
9 (x_train,y_train),(x_test,y_test)= mnist.load_data()
10 x_train=x_train.reshape(60000,28,28,1)
11 x_test=x_test.reshape(10000,28,28,1)
12 x_train=x_train.astype(np.float32)/255.0
13 x_test=x_test.astype(np.float32)/255.0
14 y_train=tf.keras.utils.to_categorical(y_train,10)
15 y_test=tf.keras.utils.to_categorical(y_test,10)
16
17 # LeNet-5 신경망 모델 설계
18 cnn=Sequential()
19 cnn.add(Conv2D(6,(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
20 cnn.add(MaxPooling2D(pool_size=(2,2)))
21 cnn.add(Conv2D(16,(5,5),padding='same',activation='relu'))
22 cnn.add(MaxPooling2D(pool_size=(2,2)))
23 cnn.add(Conv2D(120,(5,5),padding='same',activation='relu'))
24 cnn.add(Flatten())
25 cnn.add(Dense(84,activation='relu'))
26 cnn.add(Dense(10,activation='softmax'))
27
28 # 신경망 모델 학습
29 cnn.compile(loss='categorical_crossentropy',optimizer=Adam(),metrics=['accuracy'])
30 hist=cnn.fit(x_train,y_train,batch_size=128,epochs=30,validation_data=(x_test,y_test),verbose=2)
31
32 # 신경망 모델 정확률 평가
33 res=cnn.evaluate(x_test,y_test,verbose=0)
34 print("정확률은",res[1]*100)
35
36 import matplotlib.pyplot as plt
37
38 # 정확률 그래프
39 plt.plot(hist.history['accuracy'])
40 plt.plot(hist.history['val_accuracy'])
41 plt.title('Model accuracy')
42 plt.ylabel('Accuracy')
43 plt.xlabel('Epoch')
44 plt.legend(['Train','Validation'],loc='best')
45 plt.grid()
46 plt.show()
47
48 # 손실 함수 그래프
49 plt.plot(hist.history['loss'])
50 plt.plot(hist.history['val_loss'])
51 plt.title('Model loss')
52 plt.ylabel('Loss')
53 plt.xlabel('Epoch')
54 plt.legend(['Train','Validation'],loc='best')
55 plt.grid()
56 plt.show()
```

2) 실행 결과

정확률은 99.11999702453613





2. 프로그램 6-2. 컨볼루션 신경망으로 MNIST 인식

1) 프로그램 동작 설명

- 컨볼루션층을 연달아 2개 쌓는다는 측면에서 표준적인 빌딩 블록을 벗어난다.
- 전체 신경망 구조는 C-C-P-dropout-FC-droput-FC이다.
- FC층을 제외하면 LeNET-5가 C-C-P의 3개 층만 있다.

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout
from tensorflow.keras.optimizers import Adam

# MNIST 데이터셋을 읽고 신경망에 입력할 형태로 변환
(x_train,y_train),(x_test,y_test)=mnist.load_data()
x_train=x_train.reshape(60000,28,28,1)
x_test=x_test.reshape(10000,28,28,1)
x_train=x_train.astype(np.float32)/255.0
x_test=x_test.astype(np.float32)/255.0
y_train=tf.keras.utils.to_categorical(y_train,10)
y_test=tf.keras.utils.to_categorical(y_test,10)

# 신경망 모델 설계
cnn=Sequential()
cnn.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
cnn.add(Conv2D(64,(3,3),activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Dropout(0.25))
cnn.add(Flatten())
cnn.add(Dense(128,activation='relu'))
cnn.add(Dropout(0.5))
cnn.add(Dense(10,activation='softmax'))

# 신경망 모델 학습
cnn.compile(loss='categorical_crossentropy',optimizer=Adam(),metrics=['accuracy'])
hist=cnn.fit(x_train,y_train,batch_size=128,epochs=12,validation_data=(x_test,y_test),verbose=2)

# 신경망 모델 정확률 평가
res=cnn.evaluate(x_test,y_test,verbose=0)
print("정확률은",res[1]*100)

import matplotlib.pyplot as plt

# 정확률 그래프
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train','Validation'], loc='best')
plt.grid()
plt.show()

# 손실 함수 그래프
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train','Validation'], loc='best')
plt.grid()
plt.show()
```

2) 실행 결과

- 30행에서 epochs = 12로 설정해 12세대만 반복한 결과 불과 12세대만에 99.12%의 정확률을 얻었습니다.
- 프로그램 6-1의 LeNET-5보다 얕은 구조임에도 더 좋은 성능을 얻었습니다.

정확률은 99.12999868392944