딥러닝 실제 13주차 과제

2021254015 봉은정

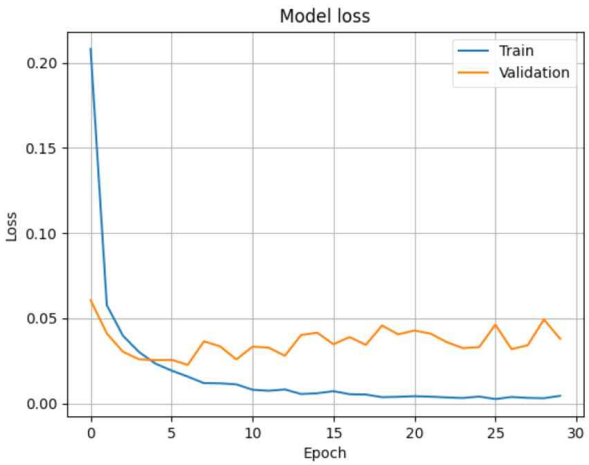
문제 1. 프로그램 6-1을 수행하여 결과를 정리하고, 프로그램의 동작을 설명하시오

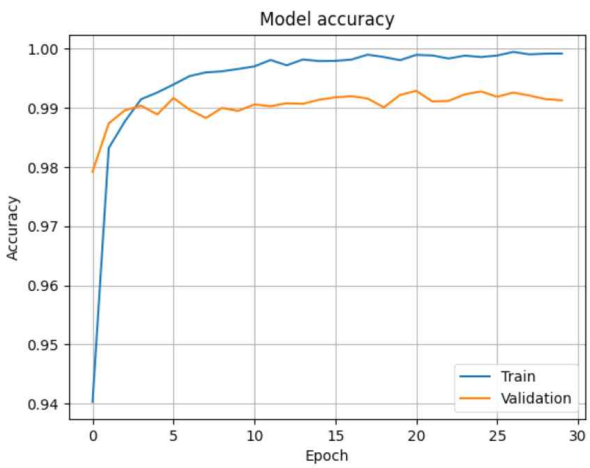
1) 수행 결과

- Train Set : epoch이 증가할수록 Loss는 감소하고, accuracy는 증가하는 경향을 보임
- Validation Set : epoch이 증가할수록 Loss는 증가하고, accuracy는 변동이 심함
- Test Set : 테스트 결과 정확도는 약 99.13%로 확인
- 오버피팅이 예측됨

```
Epoch 1/30
469/469 - 11s - loss: 0.1965 - accuracy: 0.9403 - val_loss: 0.0695 - val_accuracy: 0.9792
Epoch 2/30
469/469 - 10s - loss: 0.0539 - accuracy: 0.9832 - val_loss: 0.0415 - val_accuracy: 0.9874
Epoch 3/30
469/469 - 11s - loss: 0.0382 - accuracy: 0.9877 - val_loss: 0.0366 - val_accuracy: 0.9896
Epoch 4/30
469/469 - 11s - loss: 0.0273 - accuracy: 0.9915 - val_loss: 0.0277 - val_accuracy: 0.9904
Epoch 5/30
469/469 - 10s - loss: 0.0217 - accuracy: 0.9926 - val_loss: 0.0312 - val_accuracy: 0.9889
```

```
Epoch 26/30
469/469 - 11s - loss: 0.0036 - accuracy: 0.9989 - val_loss: 0.0374 - val_accuracy: 0.9919
Epoch 27/30
469/469 - 11s - loss: 0.0019 - accuracy: 0.9995 - val_loss: 0.0385 - val_accuracy: 0.9926
Epoch 28/30
469/469 - 11s - loss: 0.0032 - accuracy: 0.9991 - val_loss: 0.0342 - val_accuracy: 0.9921
Epoch 29/30
469/469 - 11s - loss: 0.0028 - accuracy: 0.9992 - val_loss: 0.0455 - val_accuracy: 0.9915
Epoch 30/30
469/469 - 11s - loss: 0.0025 - accuracy: 0.9992 - val_loss: 0.0452 - val_accuracy: 0.9913
정확률은 99.12999868392944
```





2) 프로그램 동작

(1) 라이브러리 import

(코드 생략)

(2) MNIST 데이터셋 불러와서 Train Set 및 Test Set으로 분할 후 정규화, 입력 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)
```

(3) 신경망 모델 설계 - Conv, ReLU, MaxPooling 반복 후 추론 단계에서 flatten, Fully Connected Layer를 거쳐 Softmax 적용

```
cnn=Sequential()
cnn.add(Conv2D(6,(5,5),padding='same',activation='relu',input_shape=(28,28,1)))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Conv2D(16,(5,5),padding='same',activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2,2)))
cnn.add(Conv2D(120,(5,5),padding='same',activation='relu'))
cnn.add(Flatten())
cnn.add(Dense(84,activation='relu'))
cnn.add(Dense(10,activation='softmax'))
```

(4) 신경망 모델 학습

- loss function : Cross Entropy / Optimizer : Adam / Batch Size = 128 / Epoch : 30

```
cnn.compile(loss='categorical_crossentropy',optimizer=Adam(),metrics=['accuracy'])
hist=cnn.fit(x_train,y_train,batch_size=128,epochs=30,validation_data=(x_test,y_test),verbose=
2)
```

(5) Test Set을 이용하여 정확도 평가

```
res=cnn.evaluate(x_test,y_test,verbose=0)
print("정확률은",res[1]*100)
```

(6) Accuracy 및 Loss Function 그래프

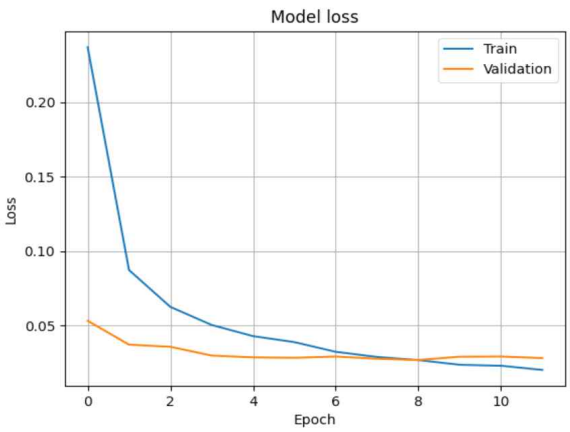
(코드 생략)

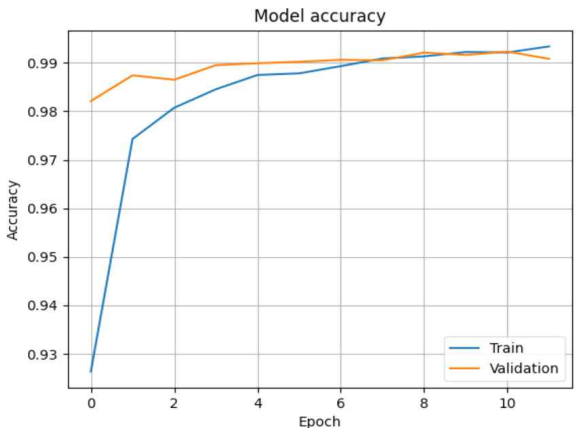
문제 2. 프로그램 6-2를 수행하여 결과를 정리하고, 프로그램의 동작을 설명하시오

1) 수행 결과

- Train Set : epoch이 증가할수록 Loss는 감소하고, accurcy는 증가하는 경향을 보임
- Validation Set : epoch이 증가할수록 Loss는 감소하고, accurcy는 증가하는 경향을 보임
- Test Set : 테스트 결과 정확도는 약 99.88%로 확인
- dropout 기법 및 적절한 epoch 설정으로, 6-1과 비교했을 때 안정적으로 학습됨을 확인

```
Epoch 1/12
469/469 - 28s - loss: 0.2368 - accuracy: 0.9263 - val_loss: 0.0531 - val_accuracy: 0.9821
Epoch 2/12
469/469 - 27s - loss: 0.0872 - accuracy: 0.9743 - val_loss: 0.0370 - val_accuracy: 0.9874
Epoch 3/12
469/469 - 28s - loss: 0.0624 - accuracy: 0.9807 - val_loss: 0.0356 - val_accuracy: 0.9865
Epoch 4/12
469/469 - 27s - loss: 0.0503 - accuracy: 0.9845 - val_loss: 0.0297 - val_accuracy: 0.9895
Epoch 5/12
469/469 - 27s - loss: 0.0428 - accuracy: 0.9875 - val_loss: 0.0285 - val_accuracy: 0.9899
Epoch 6/12
469/469 - 27s - loss: 0.0387 - accuracy: 0.9878 - val_loss: 0.0282 - val_accuracy: 0.9902
Epoch 7/12
469/469 - 27s - loss: 0.0322 - accuracy: 0.9893 - val_loss: 0.0291 - val_accuracy: 0.9906
Epoch 8/12
469/469 - 28s - loss: 0.0288 - accuracy: 0.9909 - val_loss: 0.0276 - val_accuracy: 0.9905
Epoch 9/12
469/469 - 27s - loss: 0.0267 - accuracy: 0.9913 - val_loss: 0.0267 - val_accuracy: 0.9921
Epoch 10/12
469/469 - 27s - loss: 0.0235 - accuracy: 0.9922 - val_loss: 0.0290 - val_accuracy: 0.9916
Epoch 11/12
469/469 - 28s - loss: 0.0229 - accuracy: 0.9921 - val_loss: 0.0291 - val_accuracy: 0.9923
Epoch 12/12
469/469 - 27s - loss: 0.0201 - accuracy: 0.9934 - val_loss: 0.0280 - val_accuracy: 0.9908
정확률은 99.08000230789185
```





2) 프로그램 동작

(1) 라이브러리 import

(코드 생략)

(2) MNIST 데이터셋 불러와서 Train Set 및 Test Set으로 분할 후 정규화, 입력 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)
```

(3) 신경망 모델 설계

- Conv, ReLU, MaxPooling 반복 후 추론 단계에서 flatten,
Fully Connected Layer를 거쳐 Softmax 적용. 신경망 중간에 Dropout 실행

```
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'))
```

(4) 신경망 모델 학습

- loss function : Cross Entropy / Optimizer : Adam / Batch Size = 128 / Epoch : 12

```
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)
```

(5) Test Set을 이용하여 정확도 평가

```
res=cnn.evaluate(x_test,y_test,verbose=0)
print("정확률은",res[1]*100)
```

(6) Accuracy 및 Loss Function 그래프

(코드 생략)