Tensorflow CNN 신경망 만들기

- fashion MNIST 데이터 셋을 이용한 신경망 만들기
- 개발 환경 : tf 버전 2.11 (2022/11)

학습 내용

- Fashion MNIST의 데이터 셋을 활용하여 CNN 신경망을 구축해 본다.
- 학습된 신경망 모델의 가중치를 저장하고, 이를 불러오는 방법을 알아본다.

목차

01 기본 신경망 만들기

목차로 이동하기

In [22]:

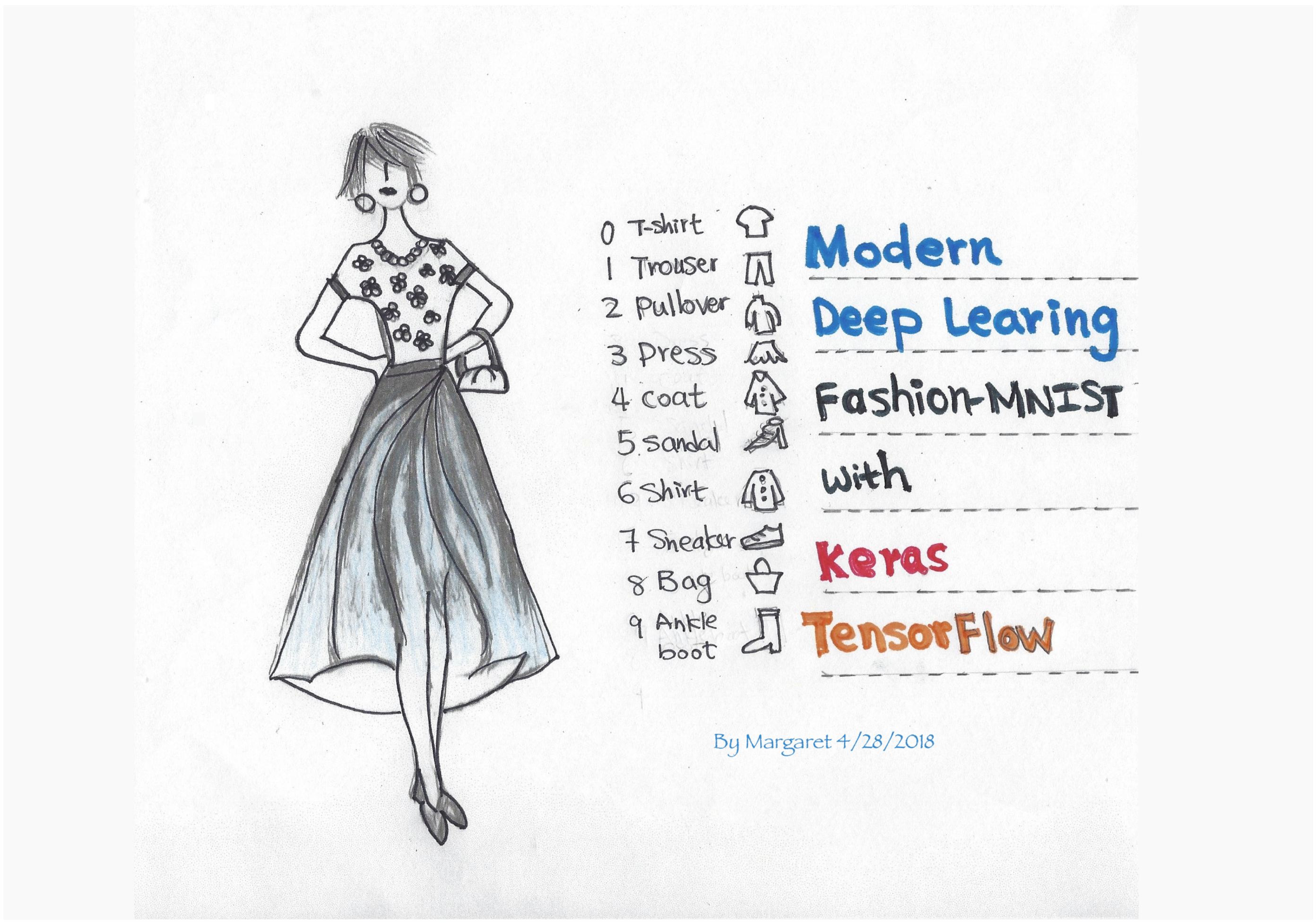
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models

import numpy as np
import matplotlib.pyplot as plt


from keras.callbacks import EarlyStopping

print(tf.__version__)
print(np.__version__)
```

```
2.11.0
1.21.5
```



In [23]:

```
fashion_mnist = keras.datasets.fashion_mnist


# 4개의 데이터 셋 반환(numpy 배열)
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

In [24]:

```
print("학습용 데이터 : x: {}, y:{}".format(X_train.shape, y_train.shape) )
print("테스트 데이터 : x: {}, y:{}".format(X_test.shape, y_test.shape) )
```

```
학습용 데이터 : x: (60000, 28, 28), y:(60000,)
테스트 데이터 : x: (10000, 28, 28), y:(10000,)
```

In [25]:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

print("학습용 데이터의 레이블 ", np.unique(y_train) )
```

```
학습용 데이터의 레이블  [0 1 2 3 4 5 6 7 8 9]
```

02 모델 구축 학습, 평가

목차로 이동하기

In [26]:

```
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense
```

In [35]:

```
model = models.Sequential()

model.add(Conv2D(32, (3, 3), padding='same',
                        activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D((2,2)) )

model.add(Conv2D(32, (3,3), padding='same',
                        strides=1, activation='relu'))
model.add(MaxPool2D(pool_size=2))

# FCL(fully connected layer)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 28, 28, 32)        320       
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 14, 14, 32)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 14, 14, 32)        9248      
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 7, 7, 32)         0         
 2D)                                                             
                                                                 
 flatten_2 (Flatten)         (None, 1568)              0         
                                                                 
 dense_4 (Dense)             (None, 256)               401664    
                                                                 
 dense_5 (Dense)             (None, 10)                2570      
                                                                 
=================================================================
Total params: 413,802
Trainable params: 413,802
Non-trainable params: 0
_________________________________________________________________
```

03 학습 결과 확인 및 저장, 불러오기

목차로 이동하기

In [36]:

```
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

In [39]:

```
os.getcwd()
```

Out[39]:

```
'D:₩₩GitHub₩₩DeepLearning_Basic_Class'
```

In [43]:

```
early_stopping = EarlyStopping(patience = 30,
                               monitor="val_loss",
                               mode="min") # 조기종료 콜백함수 정의

MODEL_SAVE_FOLDER_PATH = "./model/"

if not os.path.exists(MODEL_SAVE_FOLDER_PATH):
    os.mkdir(MODEL_SAVE_FOLDER_PATH)

model_path = MODEL_SAVE_FOLDER_PATH + "{epoch:02d}_{val_loss:.4f}.hdf5"
model_path
```

Out[43]:

```
'./model/{epoch:02d}-{val_loss:.4f}.hdf5'
```

In [44]:

```
# checkpoint_path : 모델을 저장할 경로
# monitor : 모델을 저장할 때, 기준이 되는 값
# verbose : 0 (모델이 저장될 경우, 저장), 1 (화면에 표시없이 저장)
# save_best_only : True(모니터 되는 값을 기준으로 가장 좋은 값으로 저장)
#                , False(매 에폭마타 모델이 저장.)
#  mode : 'auto', 'min', 'max'

checkpoint = ModelCheckpoint(filepath = model_path, monitor='val_loss',
                             verbose=1,
                             save_best_only=True, mode='min')

callbacks_list = [checkpoint, early_stopping]
```

In [45]:

```
%%time

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

hist = model.fit(X_train, y_train, validation_data=(X_test, y_test),
          epochs=3, batch_size=32,
          callbacks=callbacks_list )
```

```
Epoch 1/3
1874/1875 [============================>.] - ETA: 0s - loss: 0.2374 - accuracy: 0.91
25
Epoch 1: val_loss improved from inf to 0.31792, saving model to ./model₩01-0.3179.hd
f5
1875/1875 [==============================] - 64s 34ms/step - loss: 0.2374 - accurac
y: 0.9125 - val_loss: 0.3179 - val_accuracy: 0.8844
Epoch 2/3
1875/1875 [==============================] - ETA: 0s - loss: 0.2028 - accuracy: 0.92
52
Epoch 2: val_loss did not improve from 0.31792
1875/1875 [==============================] - 65s 34ms/step - loss: 0.2028 - accurac
y: 0.9252 - val_loss: 0.3345 - val_accuracy: 0.8883
Epoch 3/3
1875/1875 [==============================] - ETA: 0s - loss: 0.1840 - accuracy: 0.93
16
Epoch 3: val_loss improved from 0.31792 to 0.29546, saving model to ./model₩03-0.295
5.hdf5
1875/1875 [==============================] - 70s 37ms/step - loss: 0.1840 - accurac
y: 0.9316 - val_loss: 0.2955 - val_accuracy: 0.9000
Wall time: 3min 18s
```

사전 훈련된 weights를 불러오기

- 학습을 멈춘 시점부터 다음으로 진행하기.
- 중단된 이후의 계속 학습.
- 사전 학습된 것을 불러온 이후에 예측하기.

In [64]:

```
def create_model():
    model = models.Sequential()

    model.add(Conv2D(32, (3, 3), padding='same',
                            activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPool2D((2,2)) )

    model.add(Conv2D(32, (3,3), padding='same',
                            strides=1, activation='relu'))
    model.add(MaxPool2D(pool_size=2))

    # FCL(fully connected layer)
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(10, activation='softmax'))

    return model
```

In [66]:

```
# 기본 모델 instance를 생성
model_hdf5 = create_model()
```

In [70]:

```
model_path_hdf5 = os.getcwd() + "/model/" + "03_0.2955.hdf5"
model_path_hdf5
```

Out[70]:

```
'D:₩₩GitHub₩₩DeepLearning_Basic_Class/model/03_0.2955.hdf5'
```

In [75]:

```
model_hdf5.load_weights(model_path_hdf5)

model_hdf5.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [76]:

```
score = model_hdf5.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.29545581340789795
Test accuracy: 0.8999999761581421
```