추가실습 : CNN Fashion MNIST

- 본 파일은 GPU 런타임으로 연결됩니다.
- 경우에 따라서는 GPU 연결이 원활하지 않을 수도 있습니다.

1.환경준비

(1) 라이브러리 로딩

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random as rd

from sklearn.model_selection import train_test_split
from sklearn.metrics import *

from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from keras.backend import clear_session
from keras.optimizers import Adam
from keras.datasets import mnist, fashion_mnist
```

- 함수 만들기

```
# 학습곡선 함수
def dl_history_plot(history):
    plt.figure(figsize=(10,6))
    plt.plot(history['loss'], label='train_err')
    plt.plot(history['val_loss'], label='val_err')

    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend()
    plt.grid()
    plt.show()
```

(2) 데이터로딩

```
# 케라스 데이터셋으로 부터 fashion_mnist 불러오기
(x_train, y_train), (x_val, y_val) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

```
x_train.shape, y_train.shape
```

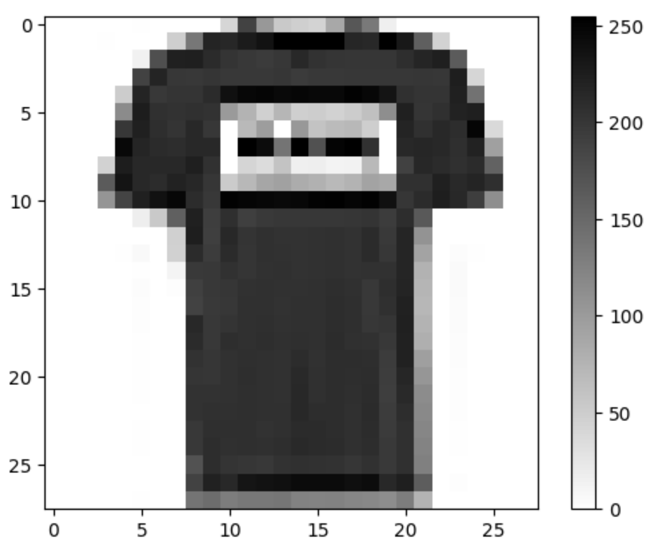
```
((60000, 28, 28), (60000,))
```

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

2 데이터 살펴보기

```
# 아래 숫자를 바꿔가며 화면에 그려 봅시다.
n = 1

plt.figure()
plt.imshow(x_train[n], cmap=plt.cm.binary)
plt.colorbar()
plt.show()
```




```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.tight_layout()
plt.show()
```

3.데이터 준비

- CNN은 3차원 구조의 이미지(데이터셋은 4차원)를 입력해야 합니다.(input_shape)

```
x_train.shape, x_val.shape
```

```
((60000, 28, 28), (10000, 28, 28))
```

- reshape를 이용하여 다음과 같이 변환해 봅시다.
  - x_train.shape : (60000, 28, 28, 1)
  - x_val.shape : (10000, 28, 28, 1)

```
x_train = x_train.reshape(60000,28,28,1)
x_val = x_val.reshape(10000,28,28,1)
```

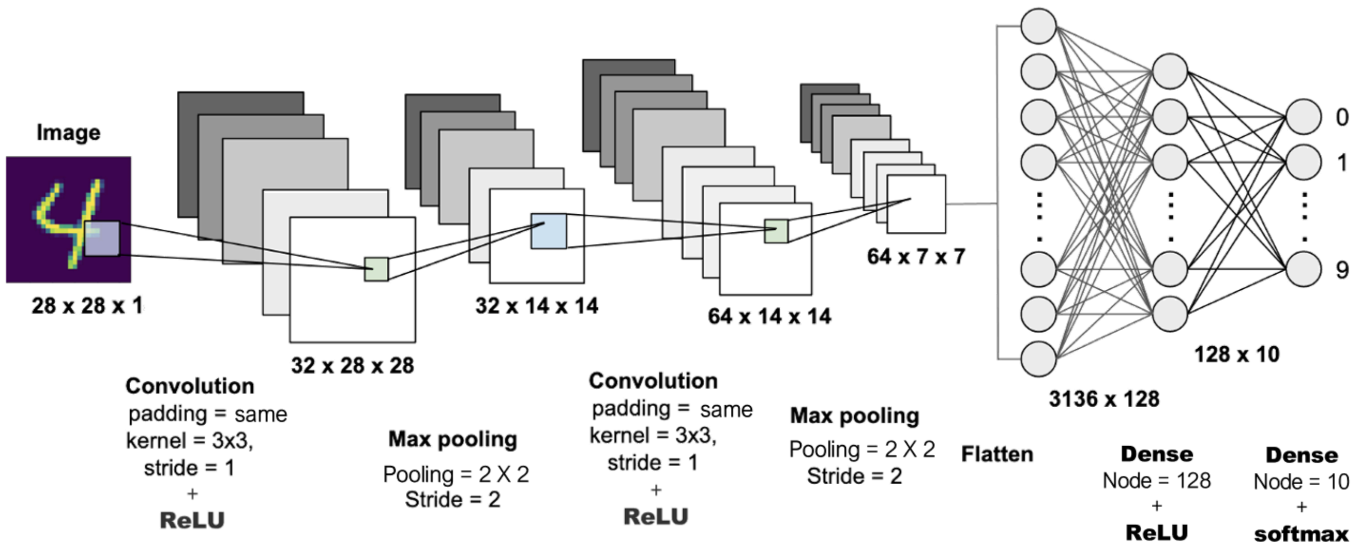
- Scaling : Min-Max

- 0-255 값으로 되어 있는 데이터를 0-1사이 값으로 변환
- x_train, x_test를 그냥 255로 나누면 됨

```
x_train = x_train / 255.
x_test = x_val / 255.
```

4.CNN 모델링

- 아래 그림의 구조대로 모델을 설계하고 학습해 봅시다.
- learning_rate = 0.0001




```
clear_session()

model = Sequential([ Conv2D(32 , kernel_size=3, input_shape=(28, 28, 1), padding='same', strides=1, activation='relu'),
                     MaxPooling2D(pool_size=2, strides=2 ),
                     Conv2D(64, kernel_size=3 ,input_shape=(14, 14, 1), padding='same', strides=1, activation='relu'),
                     MaxPooling2D(pool_size=2, strides=2 ),
                     Flatten(),
                     Dense(128, activation='relu'),
                     Dense(10, activation='softmax')  ])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2  (None, 14, 14, 32)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 64)          0         
 g2D)                                                            
                                                                 
 flatten (Flatten)           (None, 3136)              0         
                                                                 
 dense (Dense)               (None, 128)               401536    
                                                                 
 dense_1 (Dense)             (None, 10)                1290      
                                                                 
=================================================================
Total params: 421642 (1.61 MB)
Trainable params: 421642 (1.61 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

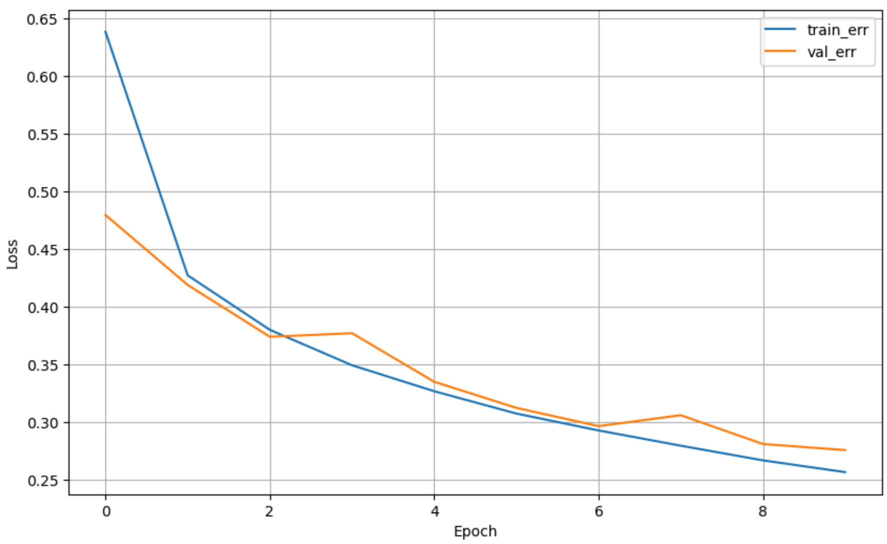
```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='sparse_categorical_crossentropy')

history = model.fit(x_train, y_train, epochs = 10,
                    validation_split=0.2).history
```

```
Epoch 1/10
1500/1500 [==============================] - 10s 4ms/step - loss: 0.6387 - val_loss: 0.4799
Epoch 2/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.4276 - val_loss: 0.4192
Epoch 3/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.3803 - val_loss: 0.3743
Epoch 4/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.3497 - val_loss: 0.3774
Epoch 5/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.3272 - val_loss: 0.3353
Epoch 6/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.3078 - val_loss: 0.3127
Epoch 7/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2932 - val_loss: 0.2969
Epoch 8/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.2800 - val_loss: 0.3064
Epoch 9/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2672 - val_loss: 0.2814
Epoch 10/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.2571 - val_loss: 0.2762
```

- 학습결과 그래프

```
dl_history_plot(history)
```



- 예측 및 평가

```
pred = model.predict(x_val)
```

```
313/313 [==============================] - 1s 2ms/step
```

```
pred_1 = pred.argmax(axis=1)
```

```
print(accuracy_score(y_val,pred_1))
print('-'*60)
print(confusion_matrix(y_val, pred_1))
print('-'*60)
print(classification_report(y_val, pred_1))
```

```
0.8431
------------------------------------------------------------
```

```
[[706   8  19   7  41   1 204   0  13   1]
 [  0 985   0   3   8   0   2   0   2   0]
 [  6   2 679   1 253   0  59   0   0   0]
 [ 11  97  12 660 171   0  46   0   3   0]
 [  0   1  16   2 967   0  13   0   1   0]
 [  0   0   0   0   0 971   1  15   1  12]
 [ 53   4  47   7 303   0 577   0   9   0]
 [  0   0   0   0   0  21   0 939   1  39]
 [  2   2   1   0  11   3   1   2 978   0]
 [  0   0   0   0   0   3   1  27   0 969]]
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.91      0.71      0.79      1000
           1       0.90      0.98      0.94      1000
           2       0.88      0.68      0.77      1000
           3       0.97      0.66      0.79      1000
           4       0.55      0.97      0.70      1000
           5       0.97      0.97      0.97      1000
           6       0.64      0.58      0.61      1000
           7       0.96      0.94      0.95      1000
           8       0.97      0.98      0.97      1000
           9       0.95      0.97      0.96      1000

    accuracy                           0.84     10000
   macro avg       0.87      0.84      0.84     10000
weighted avg       0.87      0.84      0.84     10000
```

5.틀린그림 찾아보기

위 모델의 결과에서 틀린 그림을 살펴 봅시다.

```
idx = (y_val != pred_1)
x_val_wr = x_val[idx]
y_val_wr = y_val[idx]
pred_wr = pred_1[idx]

x_val_wr = x_val_wr.reshape(-1,28,28)
print(x_val_wr.shape)
```

```
(1569, 28, 28)
```

```
idx = rd.sample(range(x_val_wr.shape[0]),25)
x_temp = x_val_wr[idx]
y_temp = y_val_wr[idx]
p_temp = pred_wr[idx]

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_temp[i], cmap=plt.cm.binary)
    plt.xlabel(f'actual : {class_names[y_temp[i]]}, \n predict : {class_names[p_temp[i]]}')
plt.tight_layout()
plt.show()
```

6.손으로 그린 그림으로 예측해 봅시다.

```
import cv2
from google.colab.patches import cv2_imshow
```

- 그림판에서 손으로 그린 그림을 업로드 합니다.

```
# 파일 열기
img = cv2.imread('coat1.png', cv2.IMREAD_GRAYSCALE)
cv2_imshow(img)
print(img.shape)
```