과목 : 인공지능과 기계학습

과제 : 손글씨 분류 구현

학번:2015039080

이름:송상호

교수:황경순 교수님

In [2]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import sys
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
import numpy as np
np.random.seed(7)

print('Python version : ', sys.version)
print('TensorFlow version : ', tf.__version__)
print('Keras version : ', keras.__version__)
```

```
Python version :  3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
TensorFlow version :  2.1.0
Keras version :  2.3.1
```

```
Using TensorFlow backend.
```

In [3]:
```
img_rows = 28
img_cols = 28

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

input_shape = (img_rows, img_cols, 1)
x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)

x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

batch_size = 128
num_classes = 10
epochs = 12

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [5]:
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1), padding='same',
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

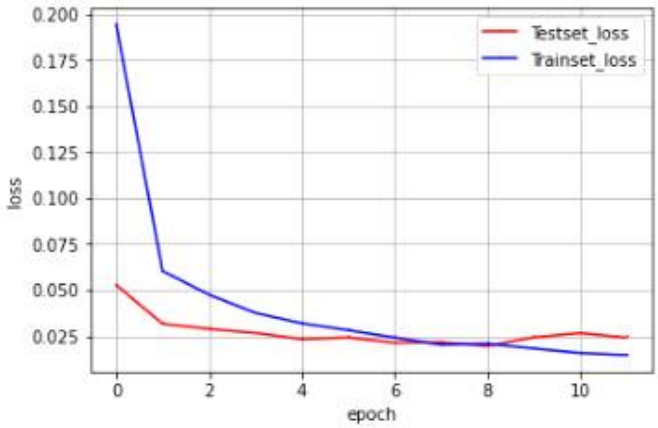
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              3137000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10010
=================================================================
Total params: 3,156,098
Trainable params: 3,156,098
Non-trainable params: 0
_________________________________________________________________
```

In [6]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
hist = model.fit(x_train, y_train,
                 batch_size=batch_size,
                 epochs=epochs,
                 verbose=1,
                 validation_data=(x_test, y_test))
```

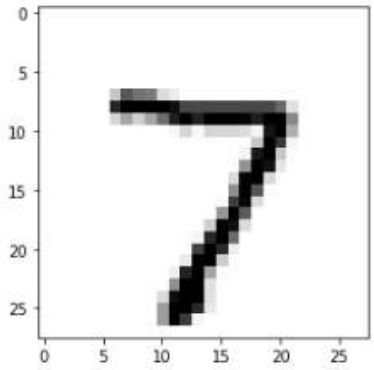
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 61s 1ms/step - loss: 0.1941 - accuracy: 0.9395 - val_loss: 0.0527 - val_accur
acy: 0.9831
Epoch 2/12
60000/60000 [==============================] - 66s 1ms/step - loss: 0.0603 - accuracy: 0.9810 - val_loss: 0.0317 - val_accur
acy: 0.9900
Epoch 3/12
60000/60000 [==============================] - 64s 1ms/step - loss: 0.0475 - accuracy: 0.9854 - val_loss: 0.0289 - val_accur
acy: 0.9908
Epoch 4/12
60000/60000 [==============================] - 63s 1ms/step - loss: 0.0375 - accuracy: 0.9880 - val_loss: 0.0267 - val_accur
acy: 0.9912
Epoch 5/12
60000/60000 [==============================] - 68s 1ms/step - loss: 0.0319 - accuracy: 0.9898 - val_loss: 0.0233 - val_accur
acy: 0.9913
Epoch 6/12
60000/60000 [==============================] - 61s 1ms/step - loss: 0.0282 - accuracy: 0.9907 - val_loss: 0.0242 - val_accur
acy: 0.9917
Epoch 7/12
60000/60000 [==============================] - 63s 1ms/step - loss: 0.0241 - accuracy: 0.9924 - val_loss: 0.0213 - val_accur
acy: 0.9933
Epoch 8/12
60000/60000 [==============================] - 63s 1ms/step - loss: 0.0203 - accuracy: 0.9931 - val_loss: 0.0216 - val_accur
acy: 0.9926
Epoch 9/12
60000/60000 [==============================] - 63s 1ms/step - loss: 0.0208 - accuracy: 0.9933 - val_loss: 0.0196 - val_accur
acy: 0.9930
Epoch 10/12
60000/60000 [==============================] - 62s 1ms/step - loss: 0.0182 - accuracy: 0.9940 - val_loss: 0.0243 - val_accur
acy: 0.9925
Epoch 11/12
60000/60000 [==============================] - 65s 1ms/step - loss: 0.0158 - accuracy: 0.9948 - val_loss: 0.0266 - val_accur
acy: 0.9915
Epoch 12/12
60000/60000 [==============================] - 62s 1ms/step - loss: 0.0146 - accuracy: 0.9952 - val_loss: 0.0241 - val_accur
acy: 0.9925
```

In [14]:
```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

import matplotlib.pyplot as plt
import numpy as np
#테스트, 학습 오차 그래프
y_vloss = hist.history['val_loss']
y_loss = hist.history['loss']
x_len = np.arange(len(y_loss))
plt.plot(x_len, y_vloss, marker=',',c="red", label='Testset_loss')
plt.plot(x_len, y_loss, marker=',',c="blue",label='Trainset_loss')
plt.legend(loc='upper right')
plt.grid()
plt.xlabel('epoch')
plt.ylabel('loss')
plt.show()
```

```
Test loss: 0.024134845009130004
Test accuracy: 0.9925000071525574
```



In [15]:
```
n = 0
plt.imshow(x_test[n].reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()

print('The Answer is ', model.predict_classes(x_test[n].reshape((1, 28, 28, 1))))
```




```
The Answer is  [7]
```