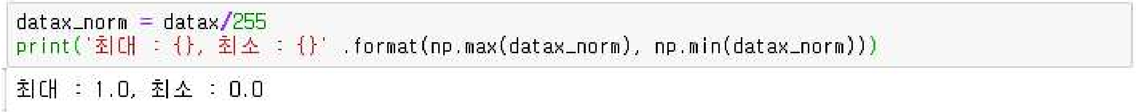
2019년 인공지능

- HW 03 -

제출일자	2019.11.20
이 름	장수훈
학 번	201402414
분 반	00

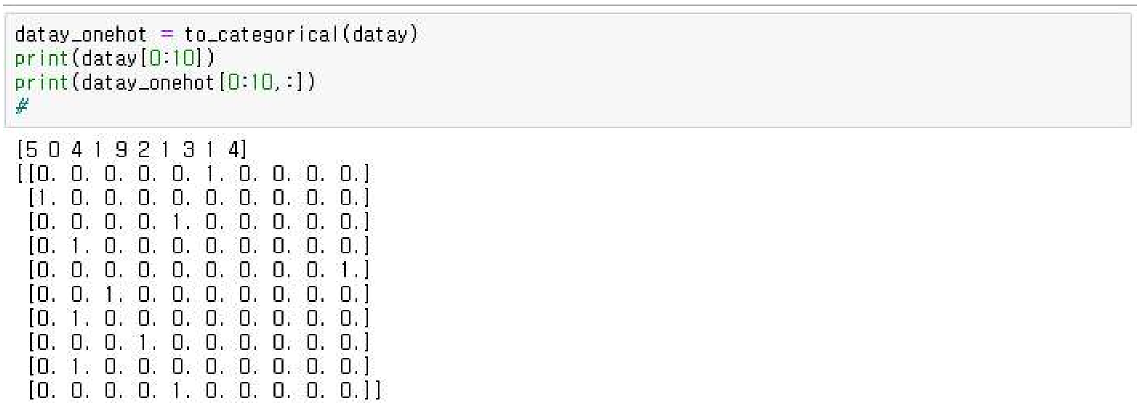
1. CNN



```
datax_norm = datax/255
print('최대 : {}, 최소 : {}' .format(np.max(datax_norm), np.min(datax_norm)))
```

```
최대 : 1.0, 최소 : 0.0
```

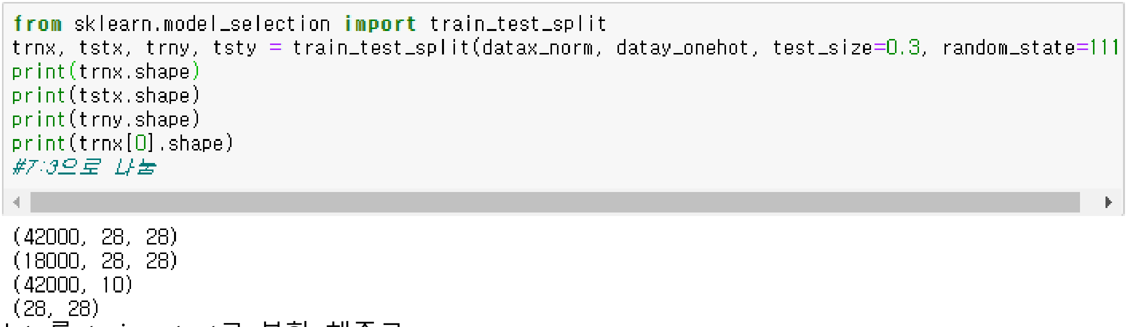
우선 정규화를 시키켜주고



```
datay_onehot = to_categorical(datay)
print(datay[0:10])
print(datay_onehot[0:10,:])
#
```

```
[5 0 4 1 9 2 1 3 1 4]
[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]
```

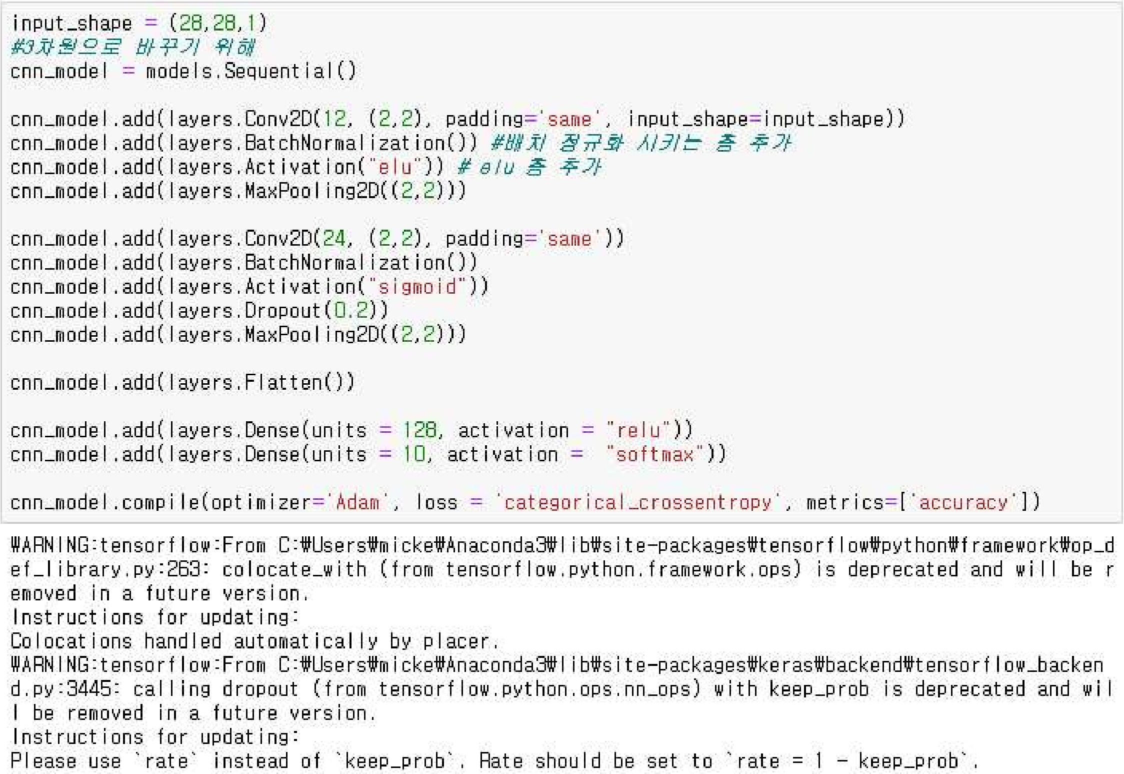
레이블이 지정된 데이터의 배열을 단일 hot벡터로 변환해주었다.



```
from sklearn.model_selection import train_test_split
trnx, tstx, trny, tsty = train_test_split(datax_norm, datay_onehot, test_size=0.3, random_state=111
print(trnx.shape)
print(tstx.shape)
print(trny.shape)
print(trnx[0].shape)
#7:3으로 나눔
```

```
(42000, 28, 28)
(18000, 28, 28)
(42000, 10)
(28, 28)
```

data를 train : test로 분할 해주고



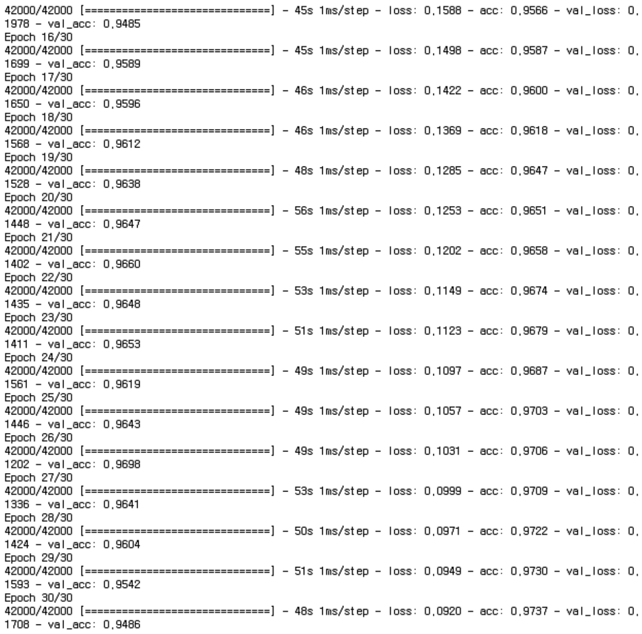
```
input_shape = (28,28,1)
#3차원으로 바꾸기 위해
cnn_model = models.Sequential()

cnn_model.add(layers.Conv2D(12, (2,2), padding='same', input_shape=input_shape))
cnn_model.add(layers.BatchNormalization()) #배치 정규화 시키는 층 추가
cnn_model.add(layers.Activation("elu")) # elu 층 추가
cnn_model.add(layers.MaxPooling2D((2,2)))

cnn_model.add(layers.Conv2D(24, (2,2), padding='same'))
cnn_model.add(layers.BatchNormalization())
cnn_model.add(layers.Activation("sigmoid"))
cnn_model.add(layers.Dropout(0.2))
cnn_model.add(layers.MaxPooling2D((2,2)))

cnn_model.add(layers.Flatten())

cnn_model.add(layers.Dense(units = 128, activation = "relu"))
cnn_model.add(layers.Dense(units = 10, activation =  "softmax"))

cnn_model.compile(optimizer='Adam', loss = 'categorical_crossentropy', metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:₩Users₩micke₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩framework₩op_d
ef_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be r
emoved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:₩Users₩micke₩Anaconda3₩lib₩site-packages₩keras₩backend₩tensorflow_backen
d.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and wil
l be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

Keras를 이용하여 CNN을 만들었다. layer설정은 주어진 예제와는 다르게 설정하였다.

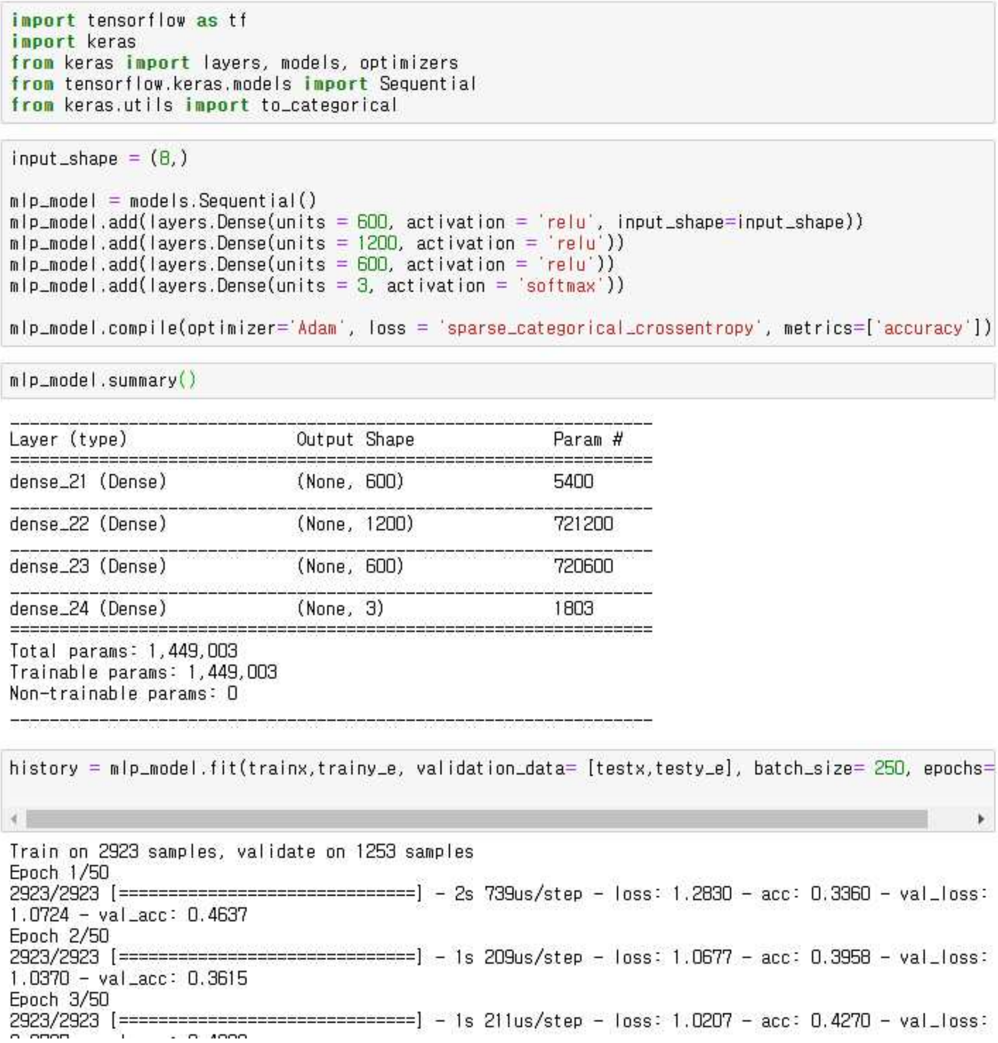
.

.



```
42000/42000 [==============================] - 45s 1ms/step - loss: 0.1588 - acc: 0.9566 - val_loss: 0.
1978 - val_acc: 0.9485
Epoch 16/30
42000/42000 [==============================] - 45s 1ms/step - loss: 0.1498 - acc: 0.9587 - val_loss: 0.
1699 - val_acc: 0.9589
Epoch 17/30
42000/42000 [==============================] - 46s 1ms/step - loss: 0.1422 - acc: 0.9600 - val_loss: 0.
1650 - val_acc: 0.9596
Epoch 18/30
42000/42000 [==============================] - 46s 1ms/step - loss: 0.1369 - acc: 0.9618 - val_loss: 0.
1568 - val_acc: 0.9612
Epoch 19/30
42000/42000 [==============================] - 48s 1ms/step - loss: 0.1285 - acc: 0.9647 - val_loss: 0.
1528 - val_acc: 0.9638
Epoch 20/30
42000/42000 [==============================] - 56s 1ms/step - loss: 0.1253 - acc: 0.9651 - val_loss: 0.
1448 - val_acc: 0.9647
Epoch 21/30
42000/42000 [==============================] - 55s 1ms/step - loss: 0.1202 - acc: 0.9658 - val_loss: 0.
1402 - val_acc: 0.9660
Epoch 22/30
42000/42000 [==============================] - 53s 1ms/step - loss: 0.1149 - acc: 0.9674 - val_loss: 0.
1435 - val_acc: 0.9648
Epoch 23/30
42000/42000 [==============================] - 51s 1ms/step - loss: 0.1123 - acc: 0.9679 - val_loss: 0.
1411 - val_acc: 0.9653
Epoch 24/30
42000/42000 [==============================] - 49s 1ms/step - loss: 0.1097 - acc: 0.9687 - val_loss: 0.
1561 - val_acc: 0.9619
Epoch 25/30
42000/42000 [==============================] - 49s 1ms/step - loss: 0.1057 - acc: 0.9703 - val_loss: 0.
1446 - val_acc: 0.9643
Epoch 26/30
42000/42000 [==============================] - 49s 1ms/step - loss: 0.1031 - acc: 0.9706 - val_loss: 0.
1202 - val_acc: 0.9698
Epoch 27/30
42000/42000 [==============================] - 53s 1ms/step - loss: 0.0999 - acc: 0.9709 - val_loss: 0.
1336 - val_acc: 0.9641
Epoch 28/30
42000/42000 [==============================] - 50s 1ms/step - loss: 0.0971 - acc: 0.9722 - val_loss: 0.
1424 - val_acc: 0.9604
Epoch 29/30
42000/42000 [==============================] - 51s 1ms/step - loss: 0.0949 - acc: 0.9730 - val_loss: 0.
1593 - val_acc: 0.9542
Epoch 30/30
42000/42000 [==============================] - 48s 1ms/step - loss: 0.0920 - acc: 0.9737 - val_loss: 0.
1708 - val_acc: 0.9486
```

성능을 확인 해 보았다.



```
import tensorflow as tf
import keras
from keras import layers, models, optimizers
from tensorflow.keras.models import Sequential
from keras.utils import to_categorical
```

```
input_shape = (8,)

mlp_model = models.Sequential()
mlp_model.add(layers.Dense(units = 600, activation = 'relu', input_shape=input_shape))
mlp_model.add(layers.Dense(units = 1200, activation = 'relu'))
mlp_model.add(layers.Dense(units = 600, activation = 'relu'))
mlp_model.add(layers.Dense(units = 3, activation = 'softmax'))

mlp_model.compile(optimizer='Adam', loss = 'sparse_categorical_crossentropy', metrics=['accuracy'])
```

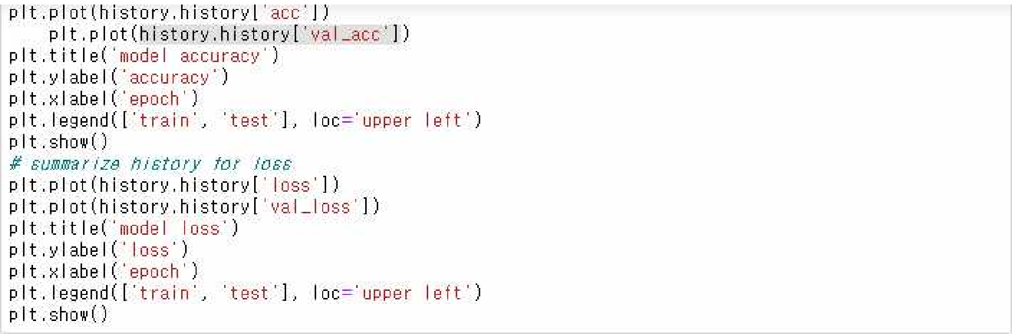
```
mlp_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_21 (Dense)             (None, 600)               5400
_________________________________________________________________
dense_22 (Dense)             (None, 1200)              721200
_________________________________________________________________
dense_23 (Dense)             (None, 600)               720600
_________________________________________________________________
dense_24 (Dense)             (None, 3)                 1803
=================================================================
Total params: 1,449,003
Trainable params: 1,449,003
Non-trainable params: 0
_________________________________________________________________
```

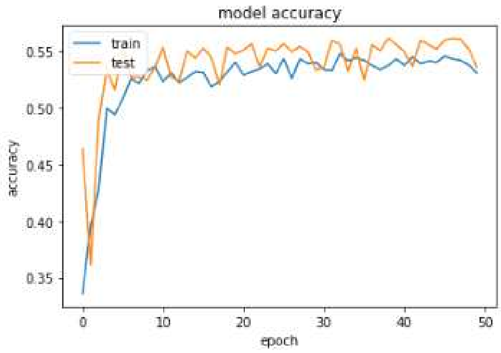
```
history = mlp_model.fit(trainx,trainy_e, validation_data= [testx,testy_e], batch_size= 250, epochs=
```

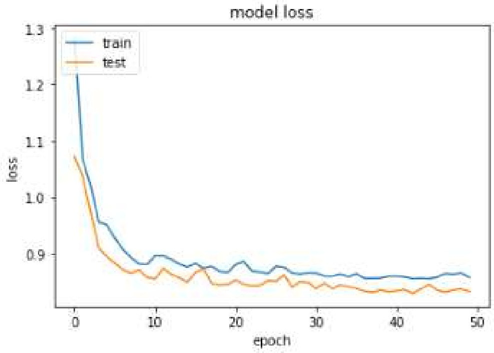
```
Train on 2923 samples, validate on 1253 samples
Epoch 1/50
2923/2923 [==============================] - 2s 739us/step - loss: 1.2830 - acc: 0.3360 - val_loss:
1.0724 - val_acc: 0.4637
Epoch 2/50
2923/2923 [==============================] - 1s 209us/step - loss: 1.0677 - acc: 0.3958 - val_loss:
1.0370 - val_acc: 0.3615
Epoch 3/50
2923/2923 [==============================] - 1s 211us/step - loss: 1.0207 - acc: 0.4270 - val_loss:
```

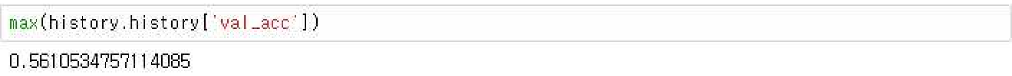
4. Keras를 이용해 Classifier B를 만들었다.



```
plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```







```
max(history.history['val_acc'])
```

```
0.5610534757114085
```

5. 정확도는 비슷하다.