

AI+X 인재양성 확대를 위한 교수자 연수

한국폴리텍대학 대구캠퍼스
SI엔지니어링학과 강현우

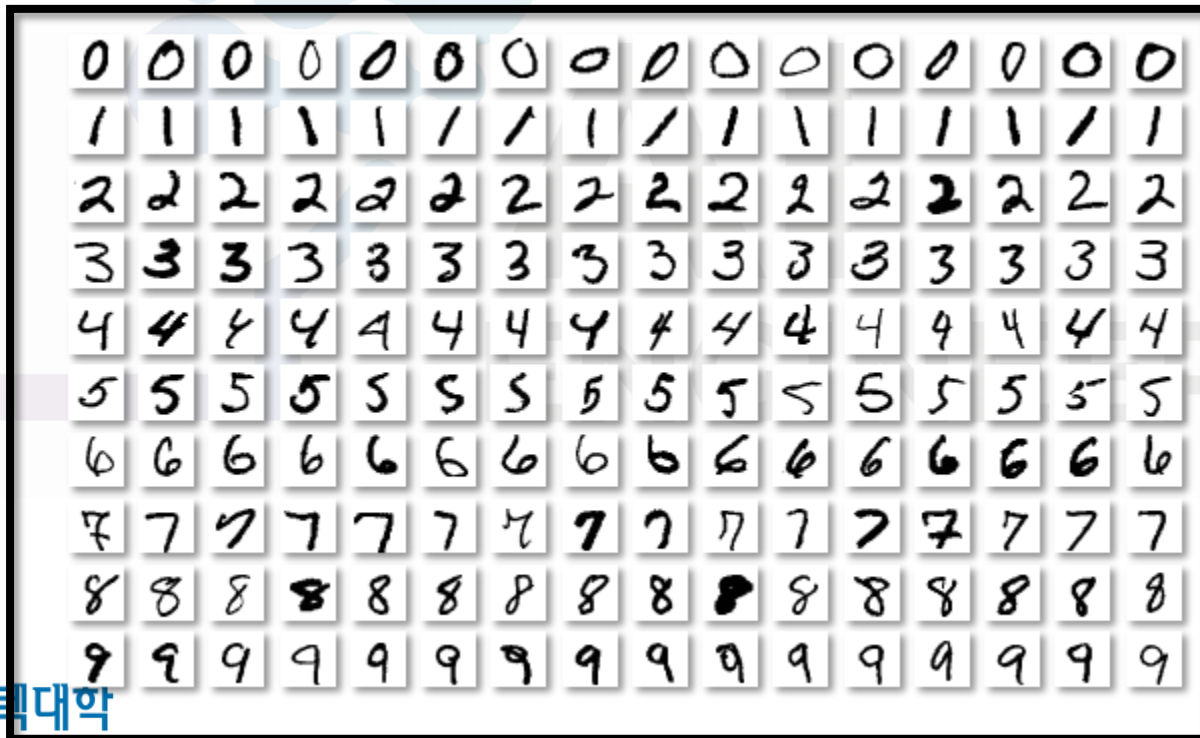
- 4강 MNIST 손글씨

- MNIST 손 글씨 데이터를 분류해본다

MNIST

◆ MNIST 데이터베이스

- NIST – 미국 국립표준기술 연구소
- National Institute of Standards and Technology
- **Modified** NIST 데이터 베이스



MNIST Data

- ◆ 손 글씨로 쓰여진 숫자 데이터

- 클래스 10개

- ◆ 데이터 셋 구성

- 트레이닝 데이터 60000개

- 테스트 데이터 10000개

- ◆ 적절한 난이도... 기계 학습 하기에 적당하다.

Data load

◆ Github 에서 다운로드 가능

➤ <https://github.com/Al-Engineering/Al-X-2021.7/tree/main/src/mnist/dataset>

```
def load_dataset(online=False):  
    if online:  
        (tr_data, tr_label), (te_data, te_label) = tf.keras.datasets.mnist.load_data()  
  
    else:  
        path = "D:/Project/Mnist/dataset/mnist.npz"  
        (tr_data, tr_label), (te_data, te_label) = tf.keras.datasets.mnist.load_data(path)  
  
    print("학습 데이터 {0}개 로드".format(tr_data.shape[0]))  
    print("테스트 데이터 {0}개 로드".format(te_data.shape[0]))  
  
    return (tr_data, tr_label), (te_data, te_label)
```

Data 확인

```
(train_set, train_label), (test_set, test_label) = load_dataset()
```

```
print(train_set.shape)
```

```
print(train_set[0].shape)
```

```
print(train_label.shape)
```

```
print(train_label[0].shape)
```

```
print(test_set.shape)
```

```
print(test_label.shape)
```

28x28 인 2차원 배열이
6만개

28x28 인 2차원 배열이
1만개

(60000, 28, 28)

(28, 28)

(60000,)

()

(10000, 28, 28)

(10000,)

ENGINEERING

Data 확인

◆ 학습 데이터

- 28x28 사이즈의 이미지
- 각 픽셀의 값은 0 ~ 255

◆ 정답 데이터

- 0 ~ 9 사이의 숫자

```
print(train_set[0])  
print(train_label[:100])
```

Data 확인

```
[[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 3 18 18 18 126 136 175 26 166 255 247 127 0 0 0 0]
[ 0 0 0 0 0 0 0 0 30 36 94 154 170 253 253 253 253 253 225 172 253 242 195 64 0 0 0 0]
[ 0 0 0 0 0 0 0 49 238 253 253 253 253 253 253 253 251 93 82 82 56 39 0 0 0 0 0]
[ 0 0 0 0 0 0 0 18 219 253 253 253 253 198 182 247 241 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 80 156 107 253 253 205 11 0 43 154 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 14 1 154 253 90 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 139 253 190 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 11 190 253 70 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 35 241 225 160 108 1 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 81 240 253 253 119 25 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 45 186 253 253 150 27 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 16 93 252 253 187 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 249 253 249 64 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 46 130 183 253 253 207 2 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 39 148 229 253 253 253 250 182 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 24 114 221 253 253 253 253 201 78 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 23 66 213 253 253 253 253 198 81 2 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 18 171 219 253 253 253 253 195 80 9 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 55 172 226 253 253 253 253 244 133 11 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 136 253 253 253 212 135 132 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

무엇 처럼 보이는가?!

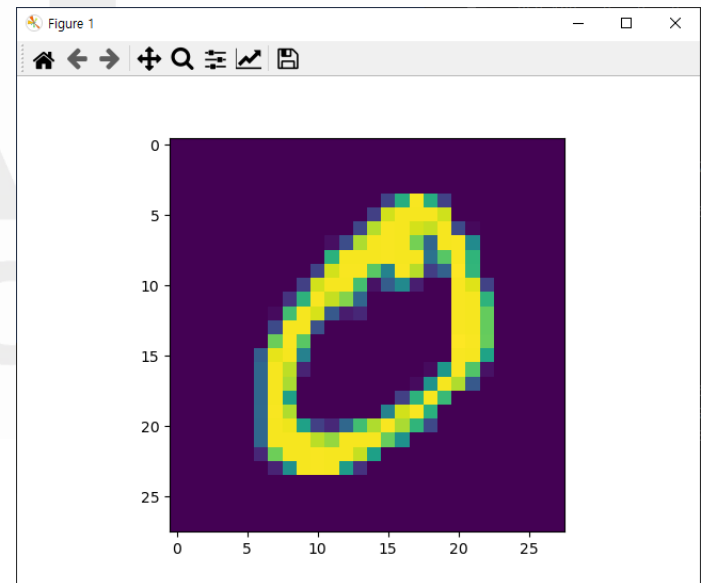
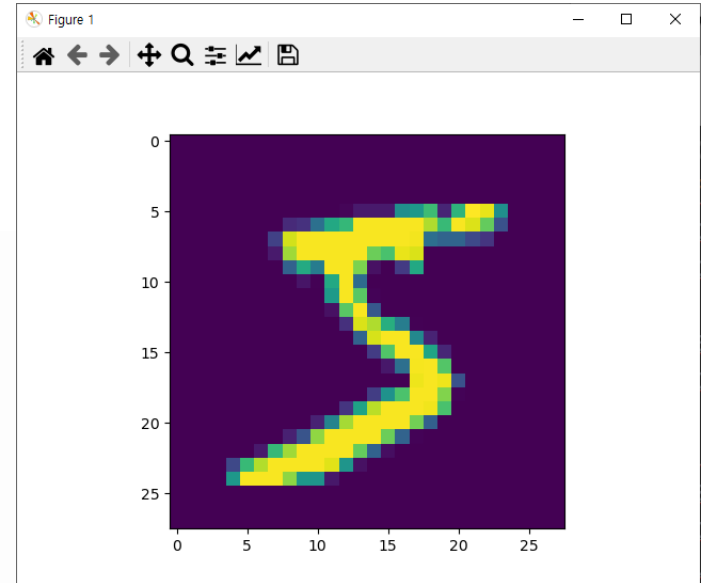
픽셀의 밝기 값을
0 ~ 255 사이로 표현

Image Show

```
# 이미지로 보고 싶을 때  
def show_image(dataset, index):  
    plt.imshow(data_set[index])  
    plt.show()
```

```
userInput = ""  
data_num = 0
```

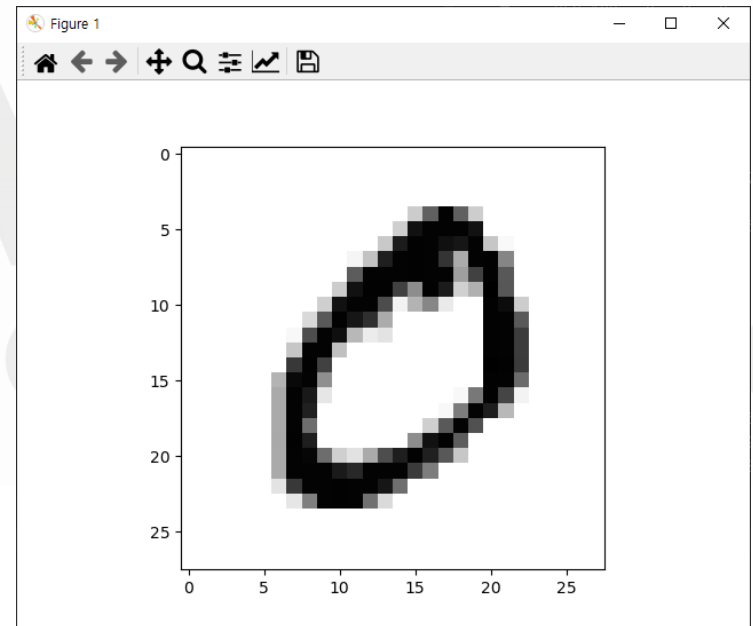
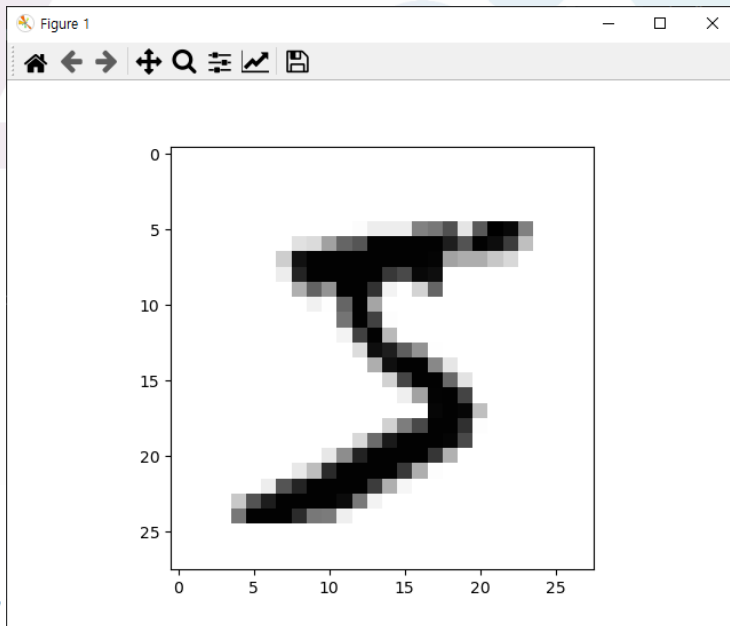
```
while userInput != "q":  
    show_image(data_num)  
    data_num += 1  
    userInput = input("next: ")
```



살짝 데코레이션

◆ 글자가 검정인 편이 익숙하니까? ㅎㅎ

```
plt.imshow(255-data_set[index], cmap="gray")
```



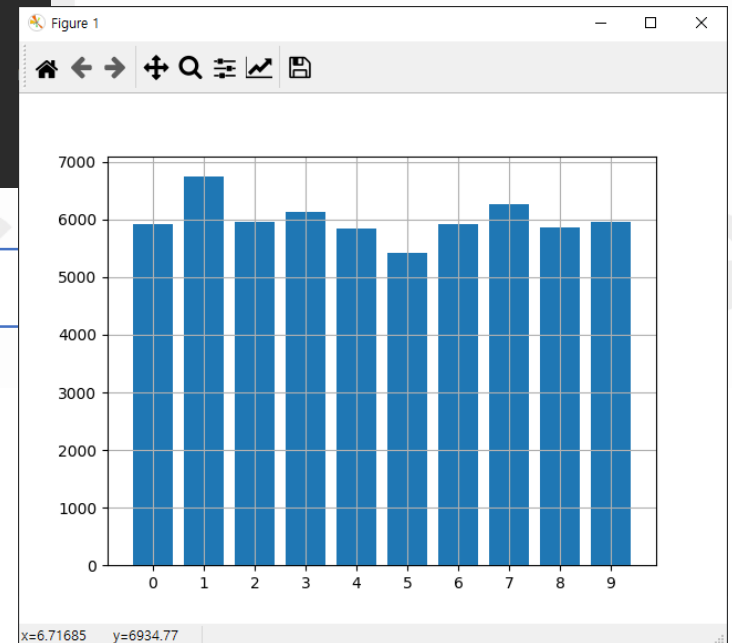
데이터 분포

<class 'numpy.ndarray'>

```
print(type(train_label))  
  
count_value = np.bincount(train_label)  
print(count_value)  
plt.bar(np.arange(0, 10), count_value)  
plt.xticks(np.arange(0, 10))  
plt.grid()  
plt.show()
```

Pandas의 value_counts
같은 것

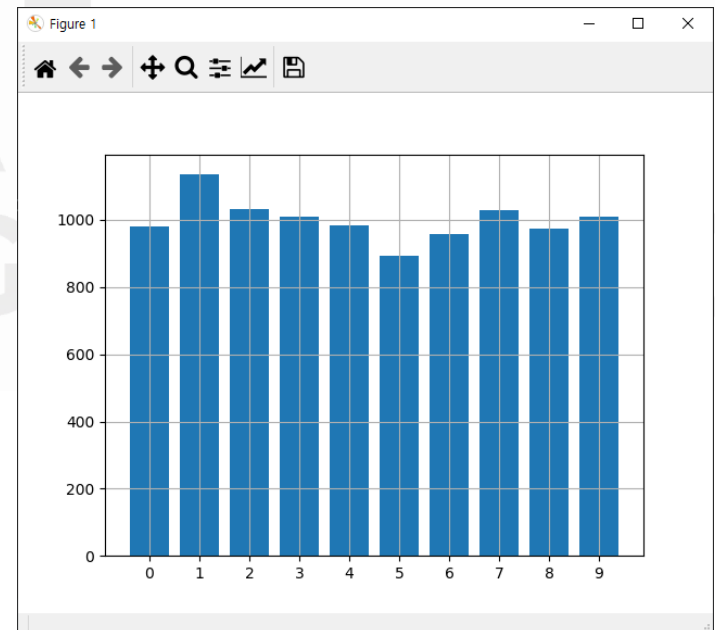
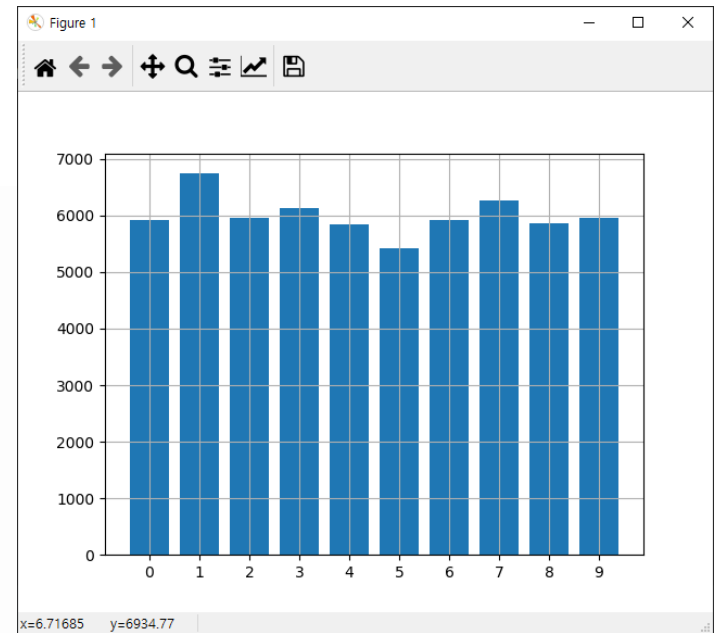
[5923 6742 5958 6131 5842 5421 5918 6265 5851 5949]



데이터 분포

```
def show_data_values(label):  
    count_value = np.bincount(label)  
    print(count_value)  
    plt.bar(np.arange(0, 10), count_value)  
    plt.xticks(np.arange(0, 10))  
    plt.grid()  
    plt.show()
```

```
# 메인  
if __name__ == "__main__":  
    (train_set, train_label), (test_set,  
    test_label) = load_dataset()  
  
    show_data_values(train_label)  
    show_data_values(test_label)
```



데이터 변환

◆ 차원 변환

- 2차원 [28 x 28] → 1차원 784로
- 0 ~ 255 uint → 0~1 float 으로

```
# 데이터 변환
```

```
train_set = train_set.reshape(60000, 784)
```

```
train_set = train_set.astype("float32")
```

```
train_set /= 255
```

```
# 짧게 쓰면 이렇게...
```

```
test_set = test_set.reshape(10000, 784).astype("float32") / 255
```

One-Hot Encoding

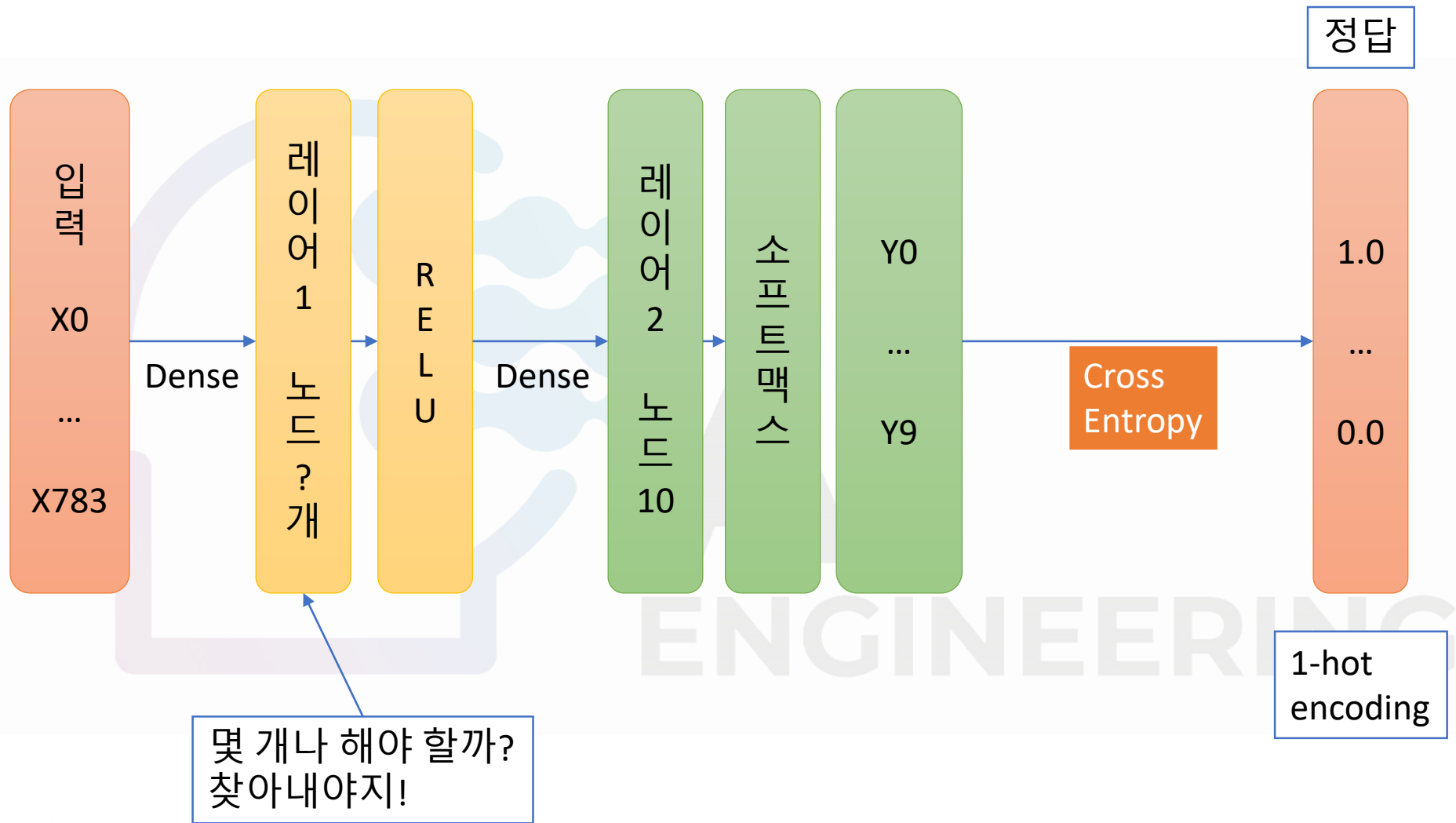
◆ 정답만 1로 나머지는 0으로..

- 1 → [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
- 5 → [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

```
# 정답은 one-hot encoding
train_label = tf.keras.utils.to_categorical(train_label, 10)
test_label = tf.keras.utils.to_categorical(test_label, 10)

print(train_label[0])
```

모델 설계



몇 개나 해야 할까?
찾아내야지!

1-hot
encoding

모델 구현

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

def make_model():
    model = Sequential()
    model.add(Dense(512, input_dim=784, activation="relu"))
    model.add(Dense(10, activation="softmax"))
    model.summary()
    model.compile(loss="categorical_crossentropy",
                  optimizer="adam",
                  metrics=["accuracy"])

    return model
```

40만 7천개 정도야... 헛

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	401920
dense_1 (Dense)	(None, 10)	5130
Total params: 407,050		
Trainable params: 407,050		
Non-trainable params: 0		

학습과 저장

```
# 하이퍼 파라미터
MY_EPOCH = 30
MY_BATCHSIZE = 200

def train(model, x, y):
    history = model.fit(x, y, epochs=MY_EPOCH, batch_size=MY_BATCHSIZE)

    model.save("./model/mlp_hd512.h5")

    return history
```

```
# 모델 생성
mlp = make_model()

# 학습
train(mlp, train_set, train_label)
```

테스트

```
# 저장된 모델로 테스트 하기 테스트
from tensorflow.keras.models import load_model
mlp = load_model("./model/mlp_hd512.h5")
acc = mlp.evaluate(test_set, test_label, batch_size=MY_BATCHSIZE)

print(acc)
```

```
50/50 [=====] - 0s 4ms/step - loss: 0.0716 -
accuracy: 0.9833
[0.07162203639745712, 0.983299970626831]
```

심화 실습

- ◆ 모델의 예측 값과 정답 레이블을 비교하여 Confusion Matrix 를 생성 하시오.

	0	1	2	3	4	5	6	7	8	9
0	98.6%	0.1%			0.4%	0.2%	0.3%	0.2%	0.1%	0.1%
1		99.4%	0.1%	0.1%		0.1%	0.2%	0.1%	0.1%	
2	0.3%		97.9%	0.1%	0.4%		0.1%	0.8%	0.5%	
3			0.5%	96.9%		0.8%	0.1%	0.4%	0.6%	0.7%
4	0.2%		0.4%		97.6%		0.3%		0.2%	1.3%
5	0.3%	0.3%		0.7%	0.1%	97.0%	0.8%	0.2%	0.4%	0.1%
6	0.3%	0.2%	0.2%		0.2%	0.4%	98.5%	0.1%		
7	0.2%	0.3%	0.9%	0.4%	0.1%	0.1%		97.2%	0.2%	0.7%
8	0.2%		0.3%	0.5%	0.4%	0.4%	0.3%	0.6%	97.0%	0.2%
9	0.1%	0.3%		0.6%	0.9%	0.5%	0.1%	0.5%	0.5%	96.5%