13주차(1/2)

텐서플로우 & 케라스

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

기계학습 오픈 프레임워크: 텐서플로우 & 케라스

- 학습 목표
  - 텐서플로우 & 케라스 **(Tensorflow & Keras)**를 이해한다.
  - **Tensorflow & Keras**를 이용하여 엠니스트**(MNIST)** 데이터를 분석한다.
  - **Tensorflow & Keras**를 이용하여 합성곱 신경망**(CNN)**을 구현한다.

- 학습 내용
  - 기계학습 오픈 프레임워크 소개
  - **Tensorflow & Keras**를 살펴본다.
  - **Tensorflow & Keras**를 이용한 **MNIST** 데이터 처리
  - **Tensorflow & Keras**를 이용한 합성곱 신경망**(CNN)** 구현

1. 기계학습 오픈 프레임워크 종류: 5가지 종류

TensorFlow

K Keras

PyTorch

scikits learn

machine learning in Python

Caffe2

Microsoft

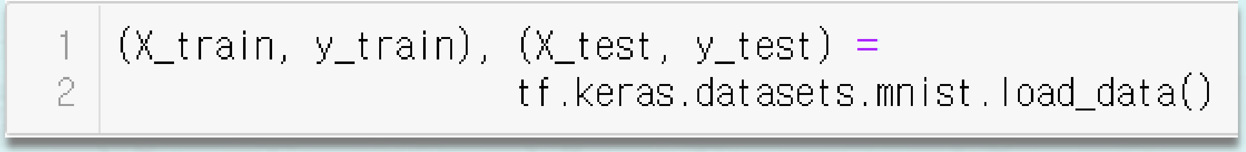
CNTK

1. 기계학습 오픈 프레임워크: 텐서플로우 & 케라스

- **TensorFlow**
  - **C++, Python** 기반
  - 합성곱 신경망**(CNN)**과 순환 신경망**(RNN)** 구현
  - **CPU, GPU** 환경 모두 동작

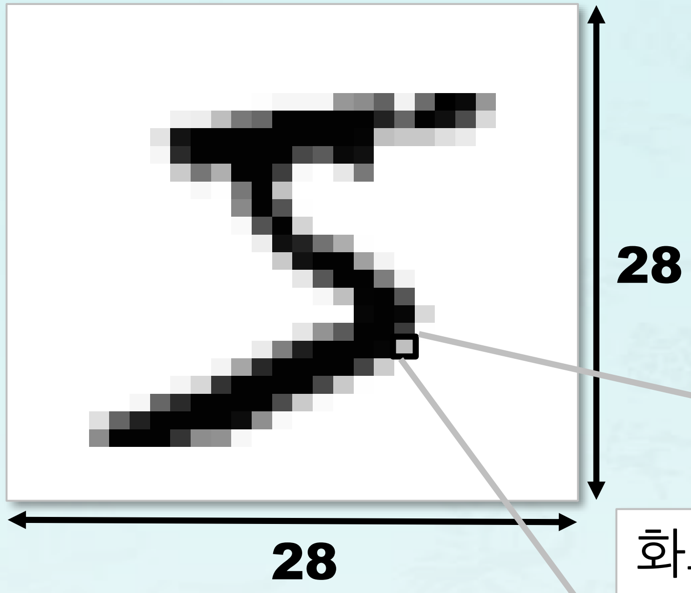
- **Keras**
  - 오픈 프레임워크 **API**
  - **Python** 기반
  - 문법이 간단하고 직관적

2. MNIST 데이터 분석: 데이터 읽어오기

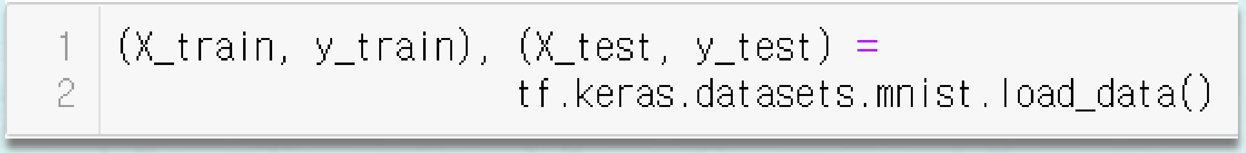


```
(X_train, y_train), (X_test, y_test) =
                    tf.keras.datasets.mnist.load_data()
```

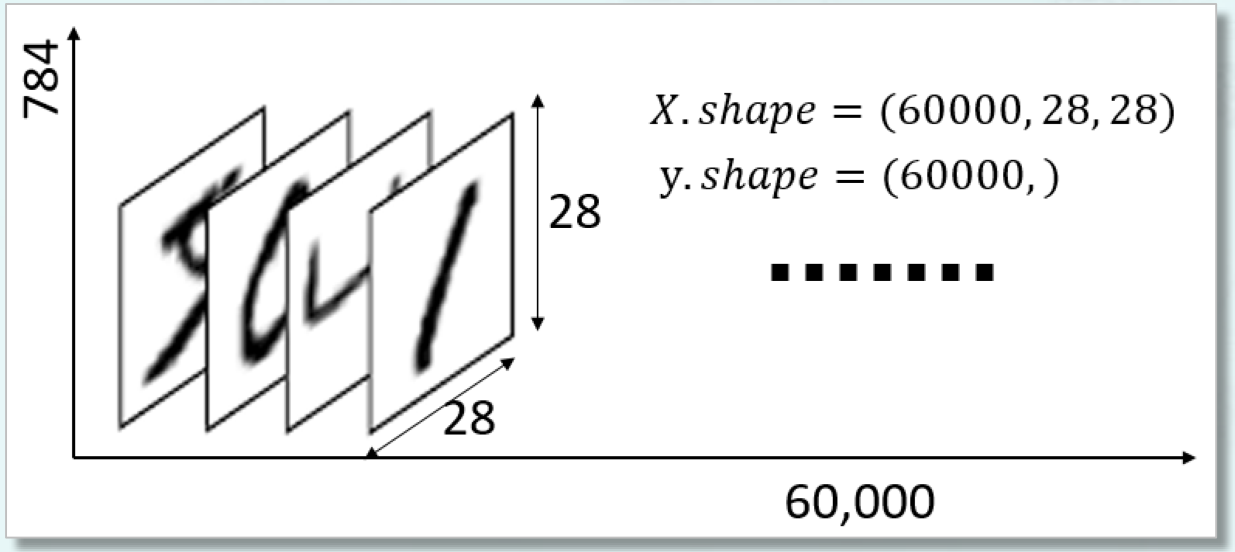
2. MNIST 데이터 분석: 데이터 읽어오기



화소(pixel) : 8bits
0(검은색) ~ 255(흰색)



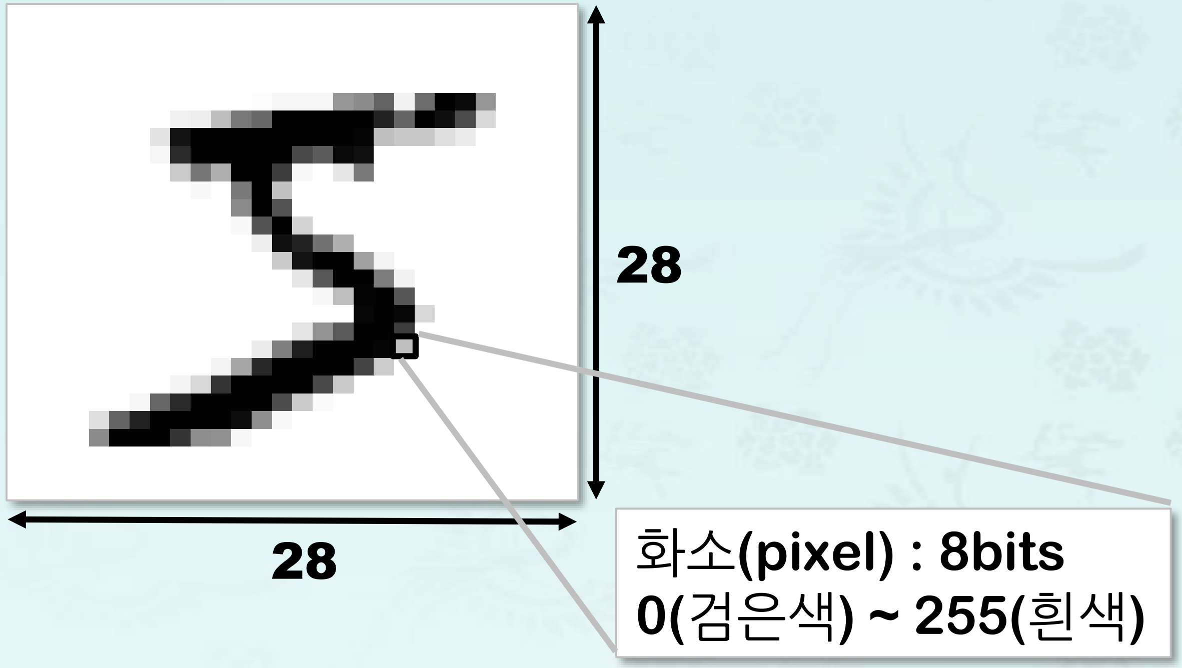
```
(X_train, y_train), (X_test, y_test) =
                        tf.keras.datasets.mnist.load_data()
```



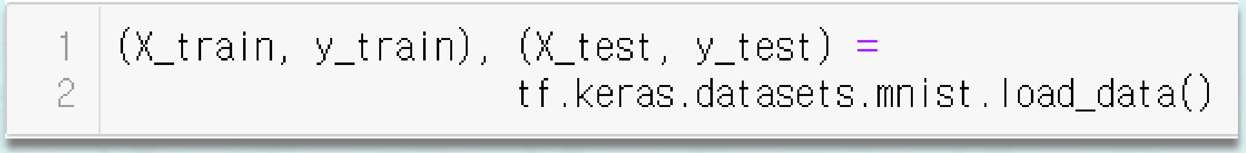
$X.shape = (60000, 28, 28)$

$y.shape = (60000,)$

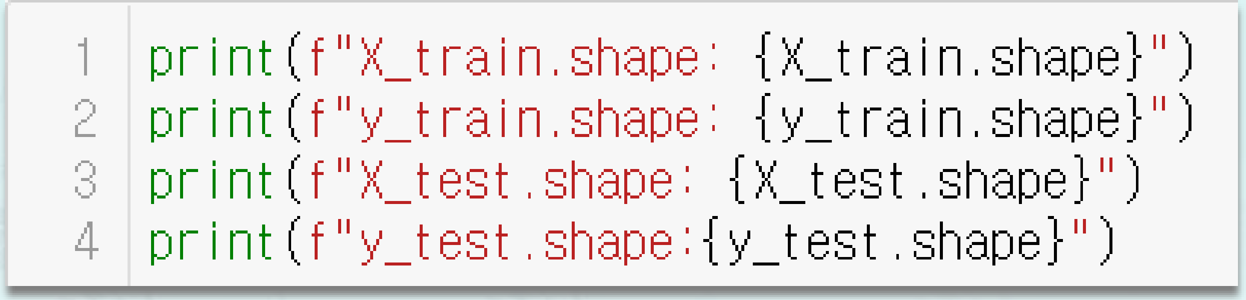
2. MNIST 데이터 분석: 데이터 읽어오기



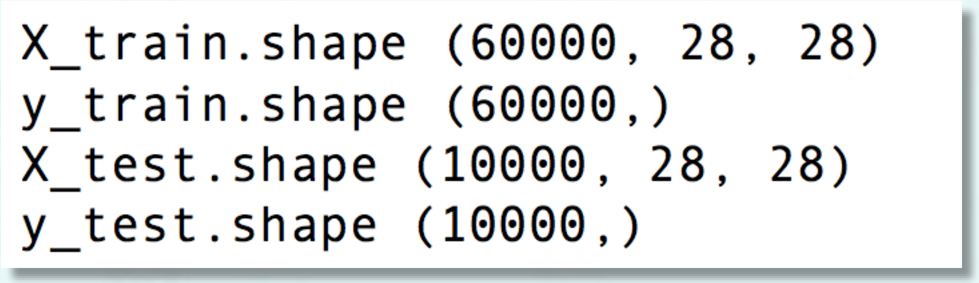
화소(pixel) : 8bits
0(검은색) ~ 255(흰색)



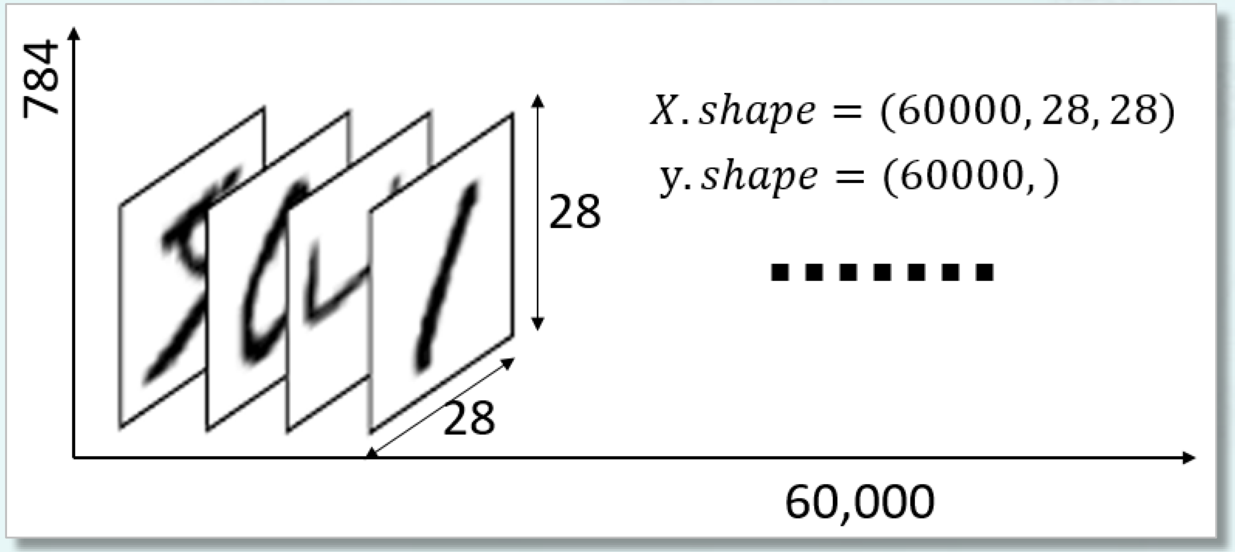
```
(X_train, y_train), (X_test, y_test) =
                    tf.keras.datasets.mnist.load_data()
```



```
print(f"X_train.shape: {X_train.shape}")
print(f"y_train.shape: {y_train.shape}")
print(f"X_test.shape: {X_test.shape}")
print(f"y_test.shape:{y_test.shape}")
```



```
X_train.shape (60000, 28, 28)
y_train.shape (60000,)
X_test.shape (10000, 28, 28)
y_test.shape (10000,)
```

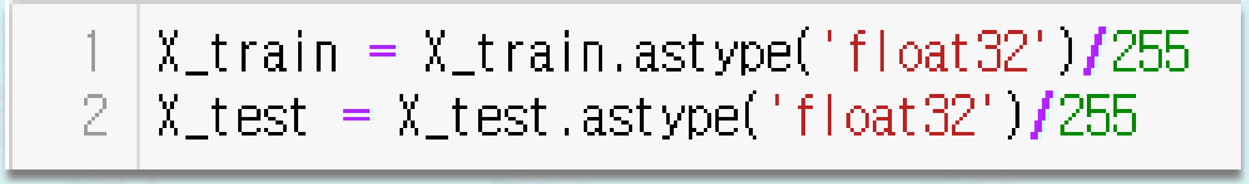


$X.shape = (60000, 28, 28)$

$y.shape = (60000,)$

2. MNIST 데이터 분석: 데이터 전처리

- 정규화(**Normalization**)

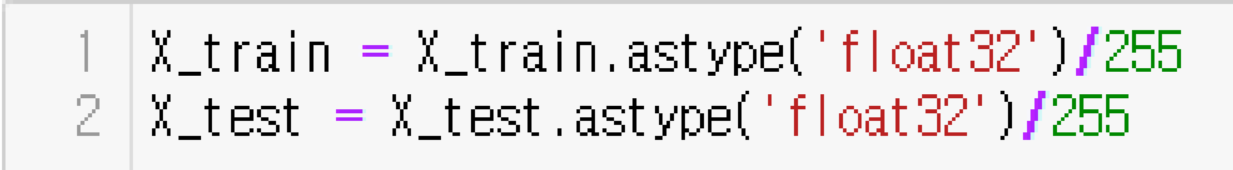


```
X_train = X_train.astype('float32')/255
X_test = X_test.astype('float32')/255
```

- **원-핫 인코딩(One-Hot Encoding)**

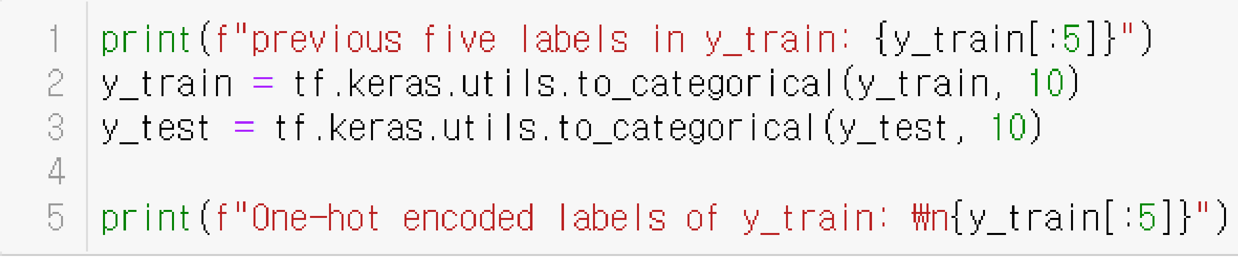
2. MNIST 데이터 분석: 데이터 전처리

- 정규화(**Normalization)**

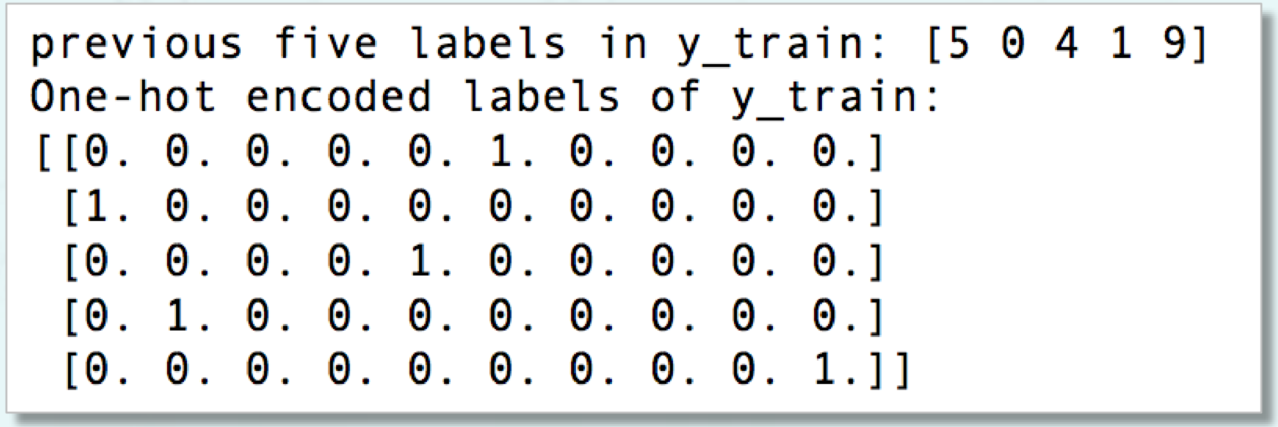


```
X_train = X_train.astype('float32')/255
X_test  = X_test.astype('float32')/255
```

- **원-핫 인코딩(One-Hot Encoding)**



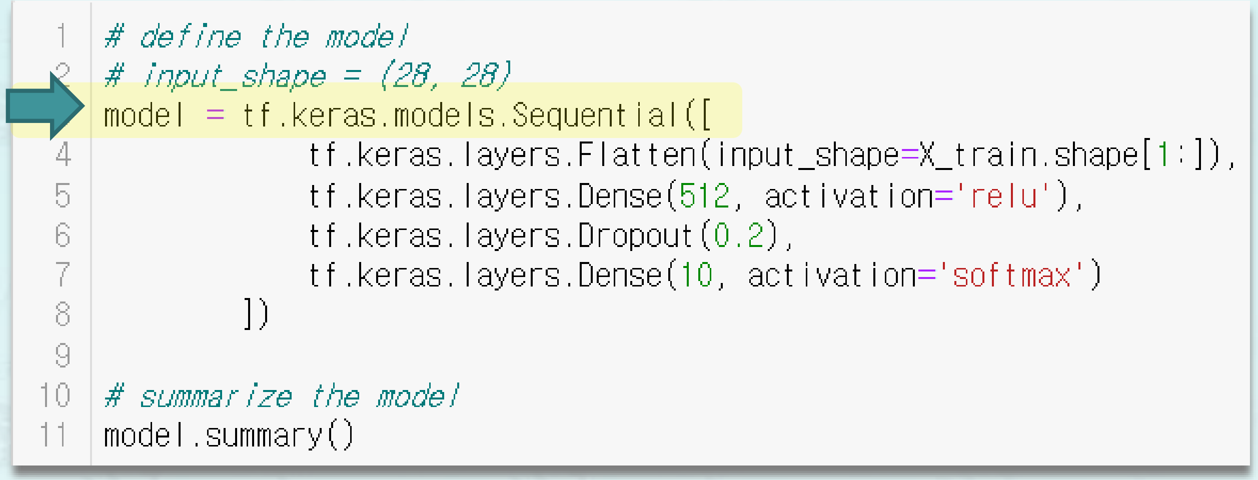
```
print(f"previous five labels in y_train: {y_train[:5]}")
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

print(f"One-hot encoded labels of y_train: ₩n{y_train[:5]}")
```



```
previous five labels in y_train: [5 0 4 1 9]
One-hot encoded labels of y_train:
[[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.]]
```

2. MNIST 데이터 분석: 신경망 구축

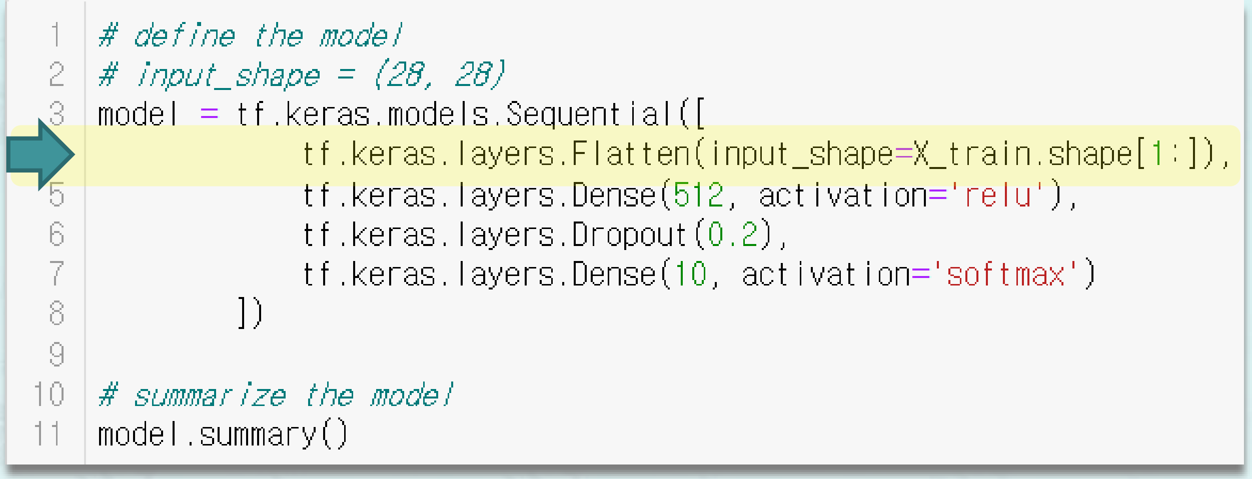
- 순차모델(**Sequential Model**)



```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```

2. MNIST 데이터 분석: 신경망 구축

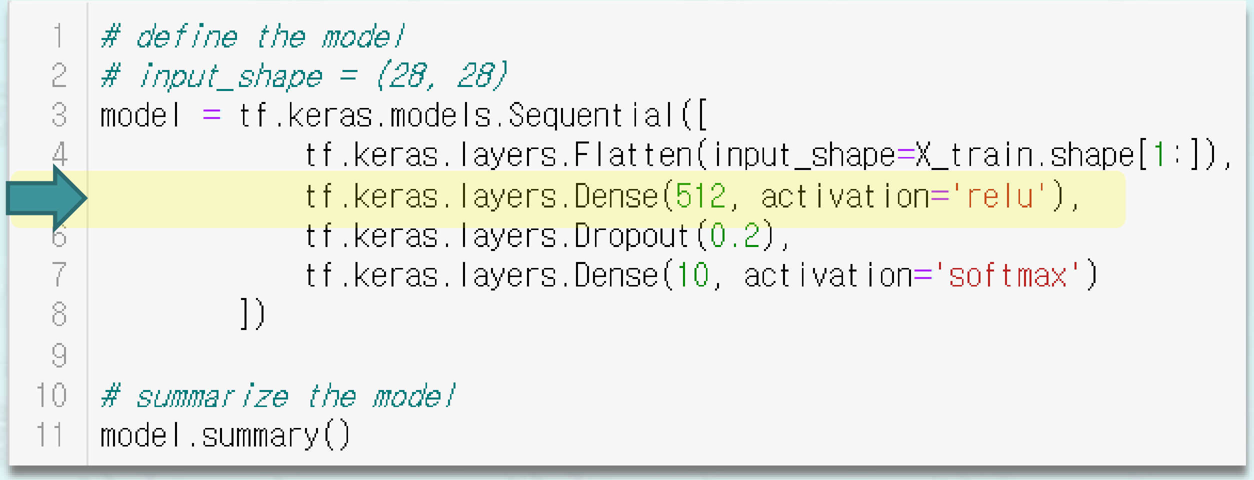
- 입력층
  - **X.shape = (60000, 28, 28)**
  - **X.shape[1:] = (28, 28)**
  - 노드(뉴론)의 수: **784 = 28 x 28**



```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```

2. MNIST 데이터 분석: 신경망 구축

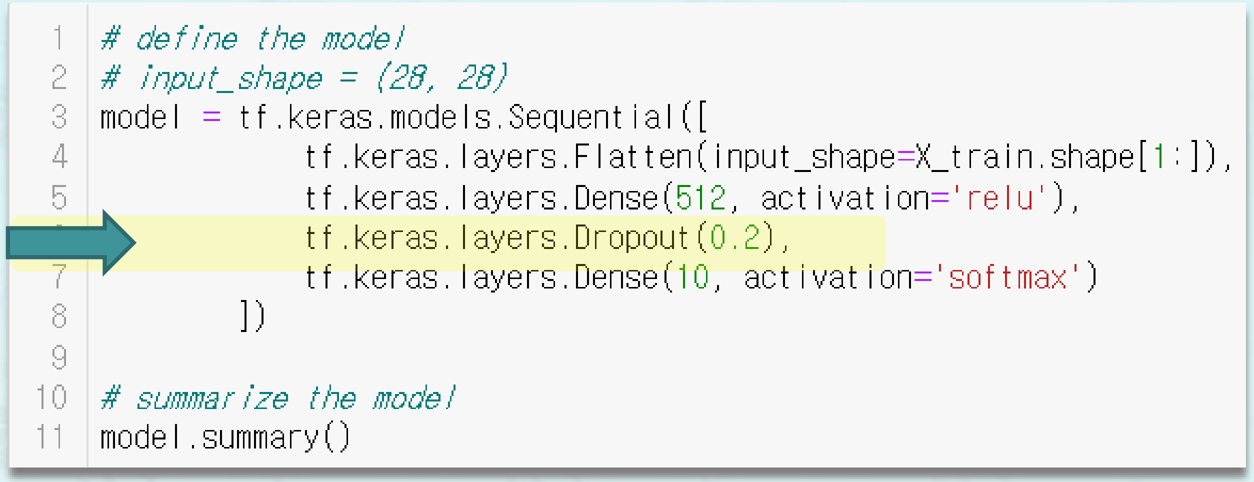
- 은닉층 – **Dense()**
  - 노드(뉴론)의 수: **512**
  - 활성화 함수: **ReLU**



```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```

2. MNIST 데이터 분석: 신경망 구축

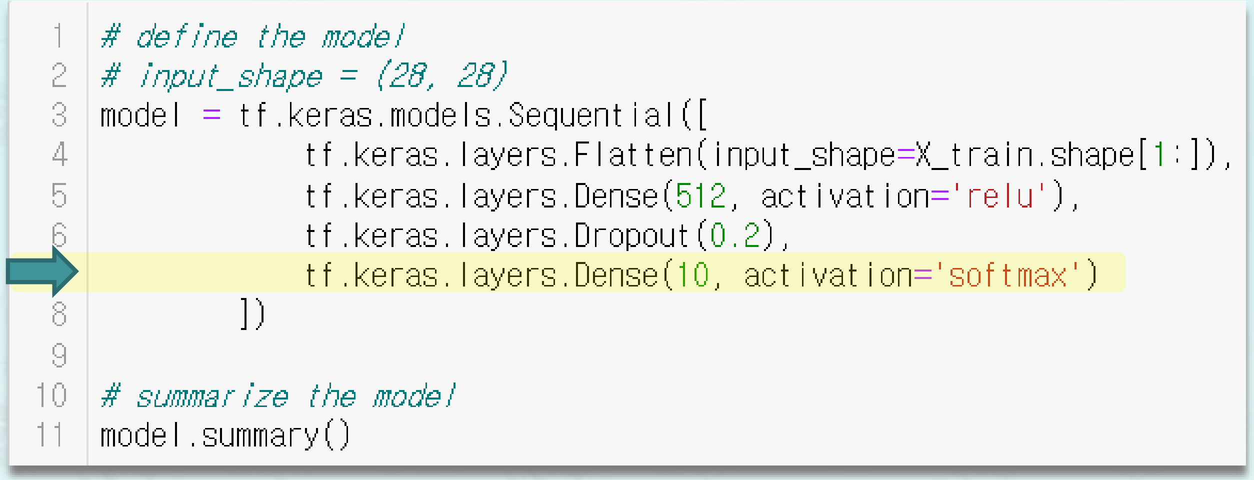
- **Dropout()**
  - 드롭아웃 비율: **0.2 (20%)**



```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```

2. MNIST 데이터 분석: 신경망 구축

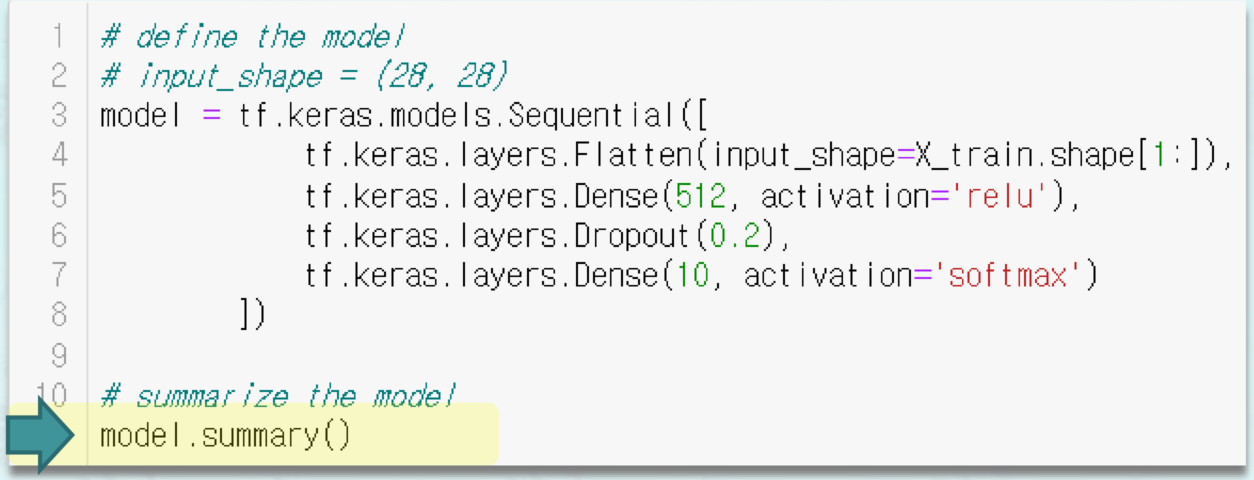
- 출력층
  - **10개 노드**
  - 활성화함수 – **softmax**



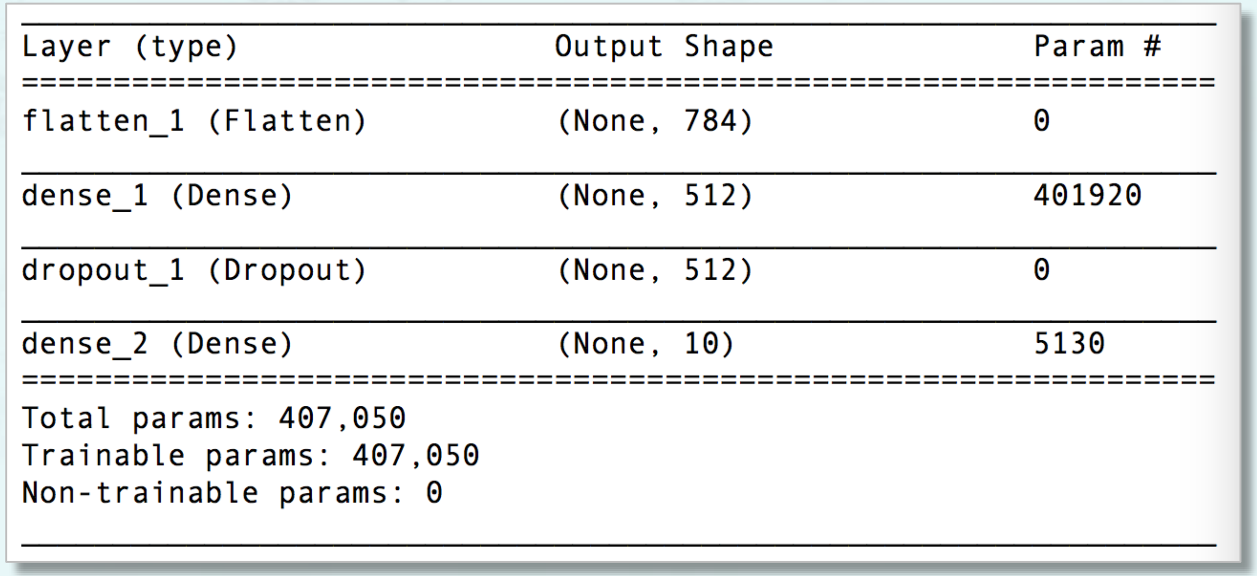
```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```

2. MNIST 데이터 분석: 신경망 구축

- **model.summary()**



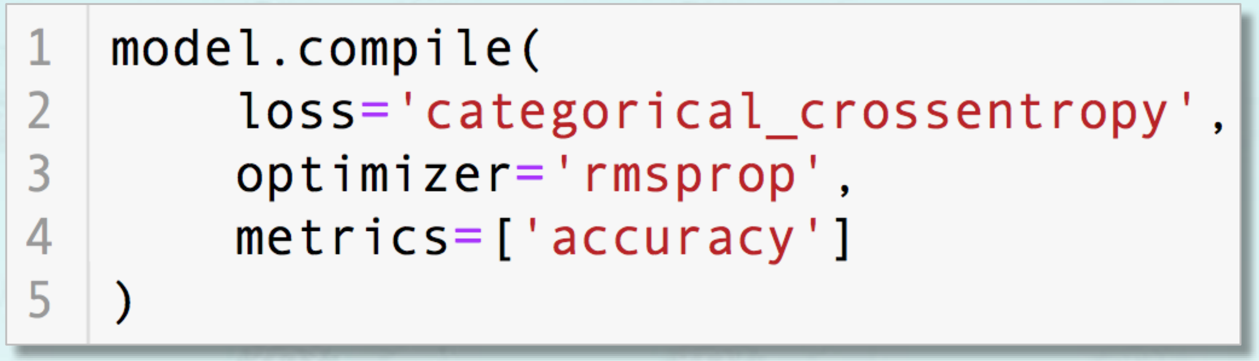
```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```



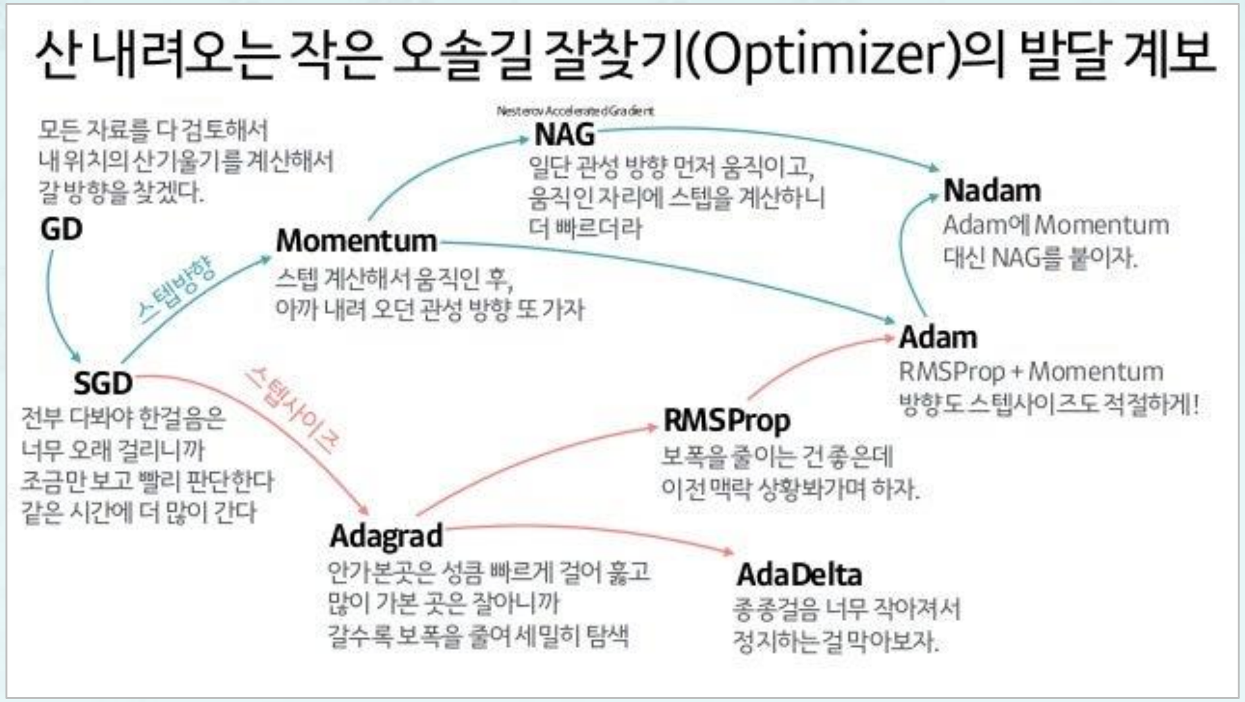
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_1 (Flatten)          (None, 784)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               401920    
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130      
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

2. MNIST 데이터 분석: 컴파일

- 손실함수**(Loss function)**
- 옵티마이저**(Optimizer)**
- 정확도**(Accuracy)**



```
model.compile(
     loss='categorical_crossentropy',
     optimizer='rmsprop',
     metrics=['accuracy']
)
```

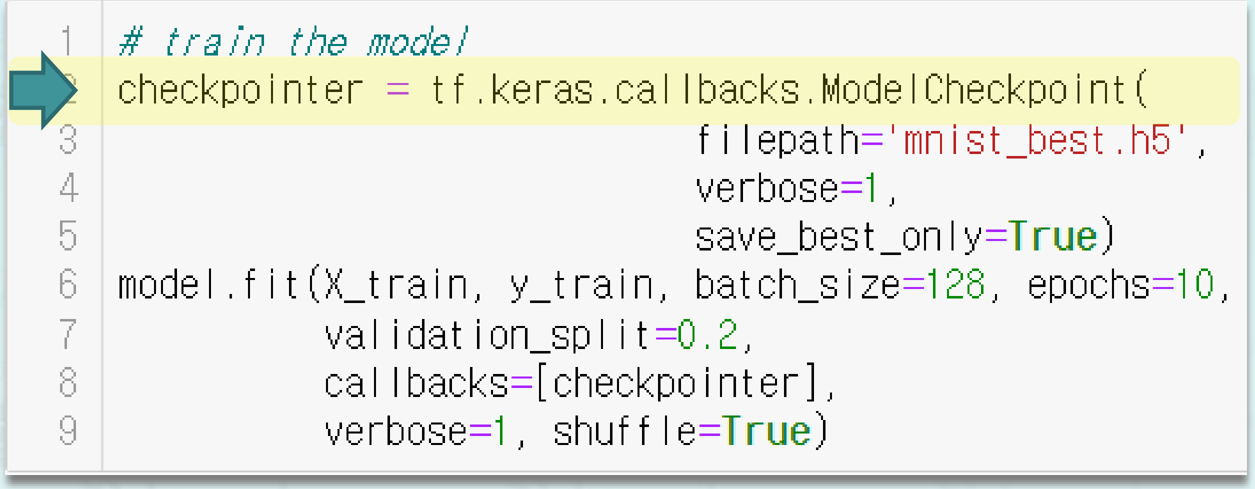


산 내려오는 작은 오솔길 잘찾기(Optimizer)의 발달 계보
모든 자료를 다 검토해서 내 위치의 산기울기를 계산해서 갈 방향을 찾겠다.
GD
스텝방향
스텝사이즈
SGD
전부 다봐야 한걸음은 너무 오래 걸리니까 조금만 보고 빨리 판단한다 같은 시간에 더 많이 간다
Momentum
스텝 계산해서 움직인 후, 아까 내려 오던 관성 방향 또 가자
Nesterov Accelerated Gradient
NAG
일단 관성 방향 먼저 움직이고, 움직인 자리에 스텝을 계산하니 더 빠르더라
Nadam
Adam에 Momentum 대신 NAG를 붙이자.
Adam
RMSProp + Momentum 방향도 스텝사이즈도 적절하게!
RMSProp
보폭을 줄이는 건 좋은데 이전 맥락 상황봐가며 하자.
Adagrad
안가본곳은 성큼 빠르게 걸어 훑고 많이 가본 곳은 잘아니까 갈수록 보폭을 줄여 세밀히 탐색
AdaDelta
종종걸음 너무 작아져서 정지하는걸 막아보자.

출처: 하용호, 자습해도 모르겠던 딥러닝

2. MNIST 데이터 분석: 모델 학습

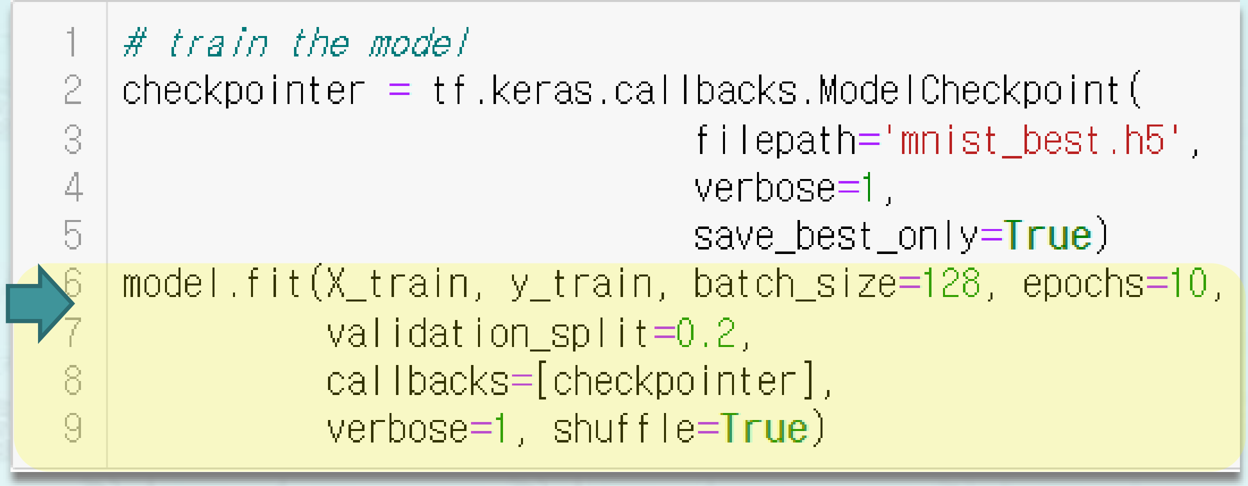
- **ModelCheckPoint()**
  - 가중치 저장 파일 **(mnist_best.h5)**



```
# train the model
checkpointer = tf.keras.callbacks.ModelCheckpoint(
                            filepath='mnist_best.h5',
                            verbose=1,
                            save_best_only=True)
model.fit(X_train, y_train, batch_size=128, epochs=10,
          validation_split=0.2,
          callbacks=[checkpointer],
          verbose=1, shuffle=True)
```

2. MNIST 데이터 분석: 모델 학습

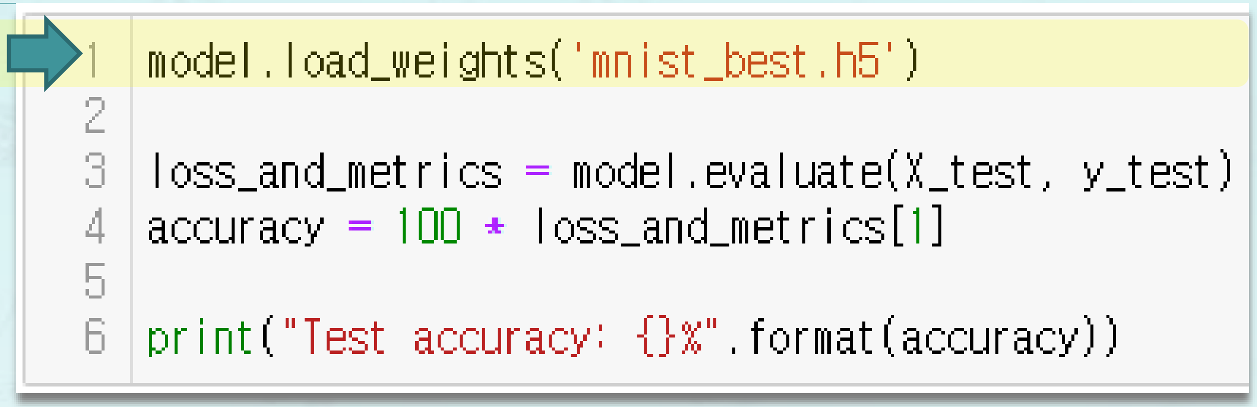
- **fit() –** 학습 파라미터 정하기



```
# train the model
checkpointer = tf.keras.callbacks.ModelCheckpoint(
                            filepath='mnist_best.h5',
                            verbose=1,
                            save_best_only=True)
model.fit(X_train, y_train, batch_size=128, epochs=10,
          validation_split=0.2,
          callbacks=[checkpointer],
          verbose=1, shuffle=True)
```

2. MNIST 데이터 분석: 분류 정확도 측정

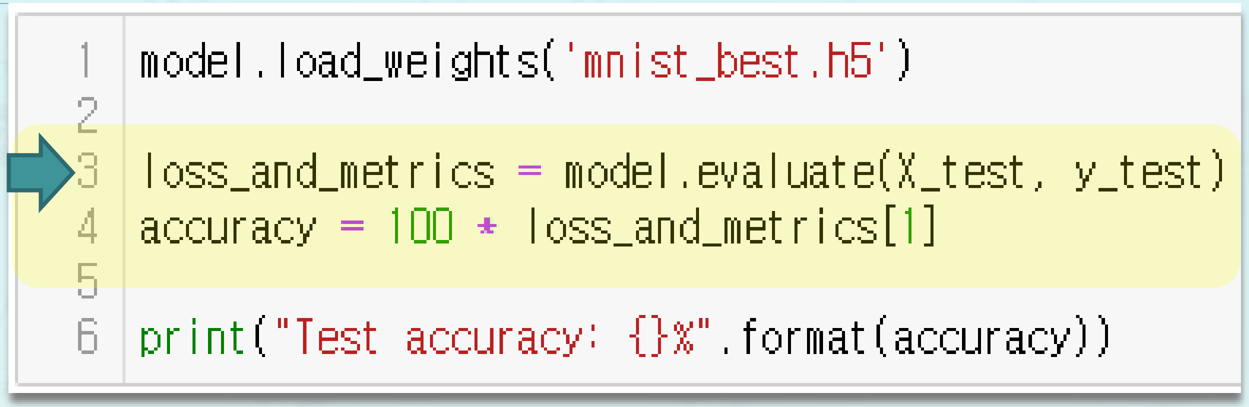
- 학습된 모델 가중치 **(mnist_best.h5)** 사용하기



```
model.load_weights('mnist_best.h5')

loss_and_metrics = model.evaluate(X_test, y_test)
accuracy = 100 * loss_and_metrics[1]

print("Test accuracy: {}%".format(accuracy))
```

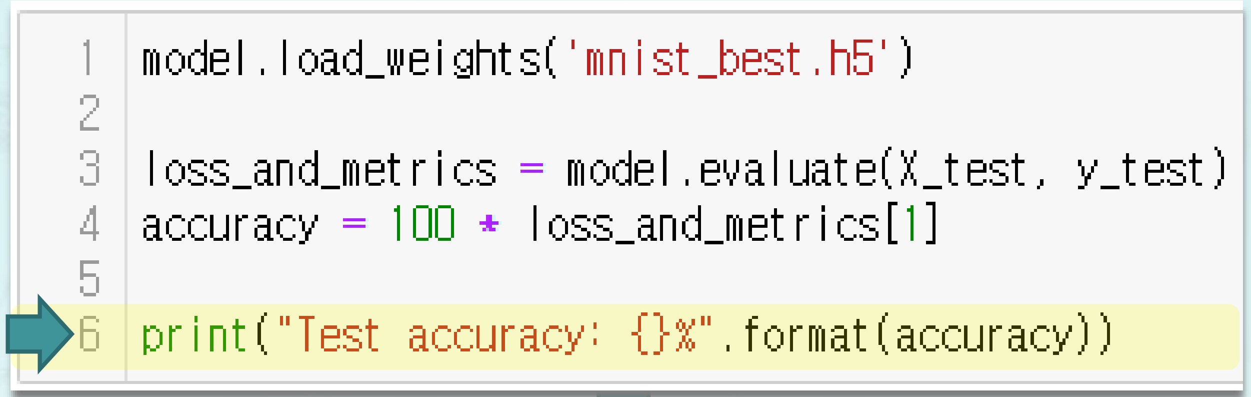
2. MNIST 데이터 분석: 분류 정확도 측정

- 평가 데이터 분류하기

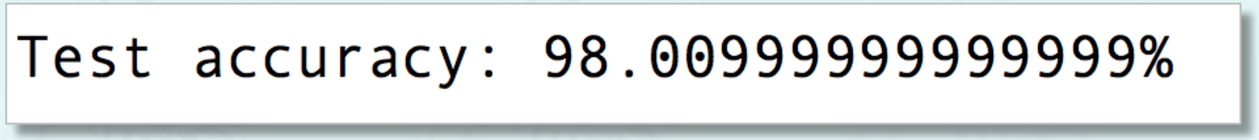


```
model.load_weights('mnist_best.h5')

loss_and_metrics = model.evaluate(X_test, y_test)
accuracy = 100 * loss_and_metrics[1]

print("Test accuracy: {}%".format(accuracy))
```

2. MNIST 데이터 분석: 분류 정확도 측정



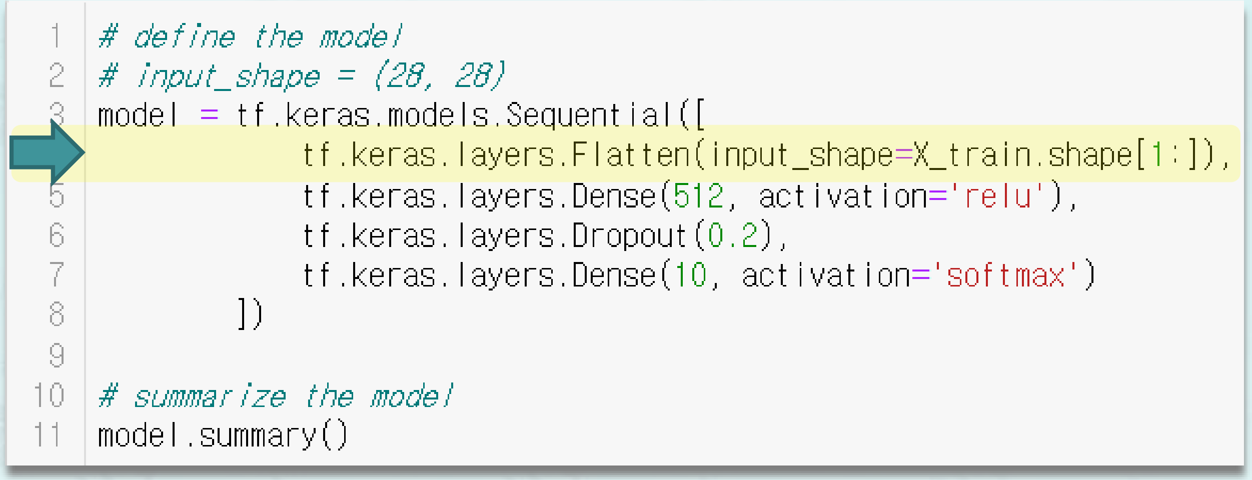
```
model.load_weights('mnist_best.h5')

loss_and_metrics = model.evaluate(X_test, y_test)
accuracy = 100 * loss_and_metrics[1]

print("Test accuracy: {}%".format(accuracy))
```



```
Test accuracy: 98.00999999999999%
```

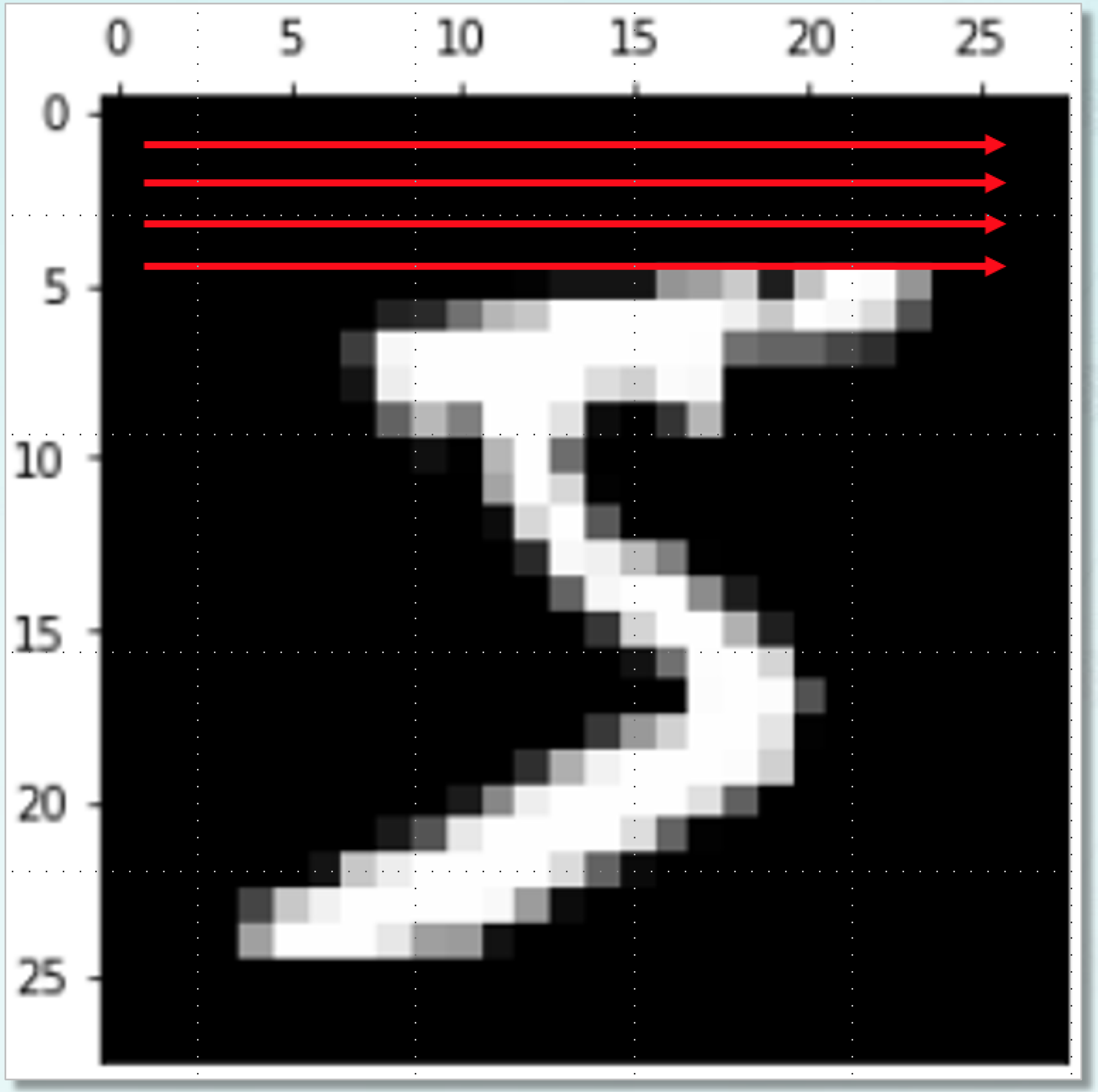
3. CNN 구현: 인공 신경망(ANN)의 한계

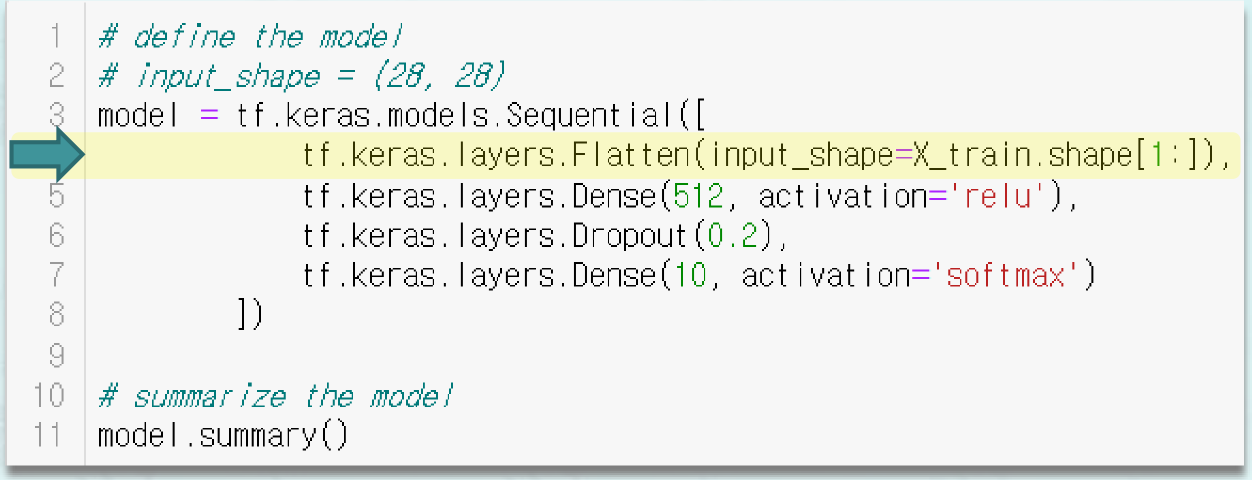
- 입력층
  - **X.shape = (60000, 28, 28)**
  - **X.shape[1:] = (28, 28)**
  - 노드(뉴론)의 수: **784 = 28 x 28**



```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
                tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
                tf.keras.layers.Dense(512, activation='relu'),
                tf.keras.layers.Dropout(0.2),
                tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```

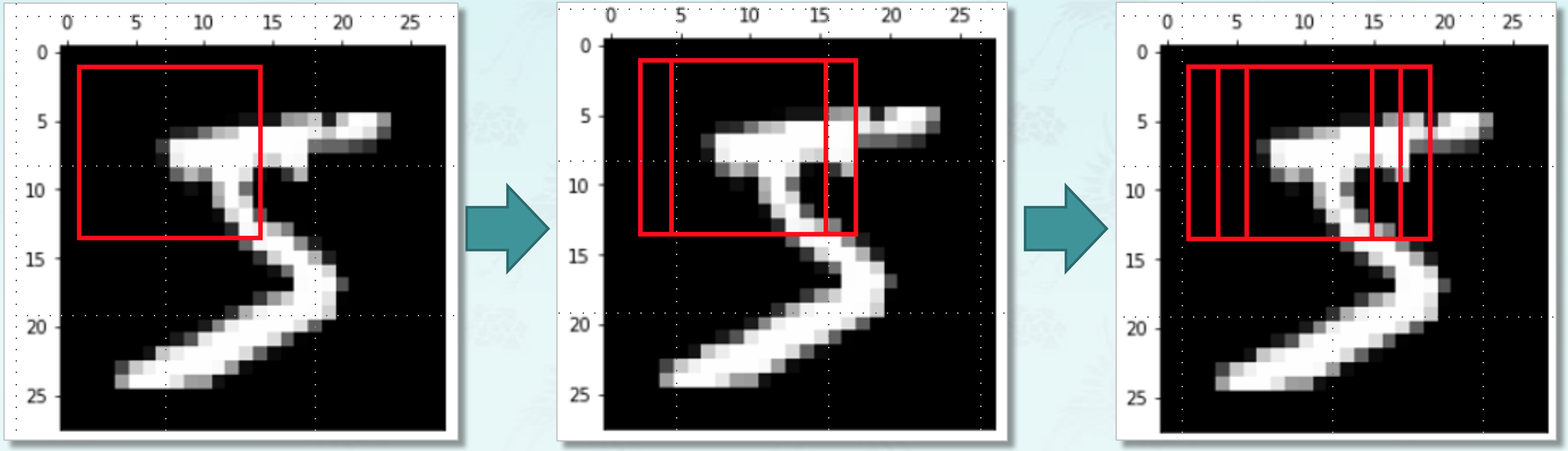
3. CNN 구현: 인공 신경망(ANN)의 한계





```
# define the model
# input_shape = (28, 28)
model = tf.keras.models.Sequential([
            tf.keras.layers.Flatten(input_shape=X_train.shape[1:]),
            tf.keras.layers.Dense(512, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(10, activation='softmax')
        ])

# summarize the model
model.summary()
```

3. CNN 구현: 합성곱층(Convolutional Layer)



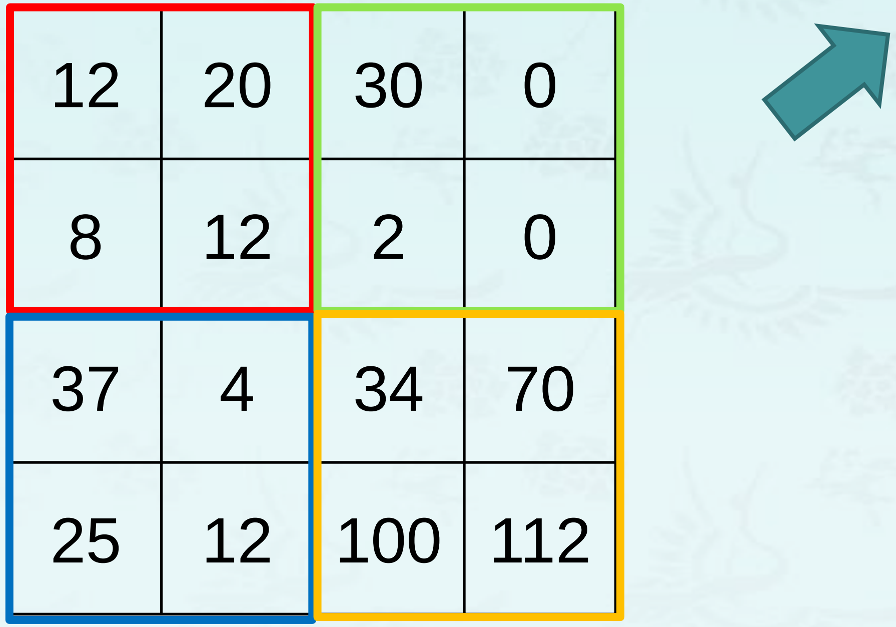
3. CNN 구현: Pooling Layer

12	20	30	0
8	12	2	0
37	4	34	70
25	12	100	112

- **Max Pooling Layer**

- **Global Average Pooling Layer**

3. CNN 구현: Pooling Layer



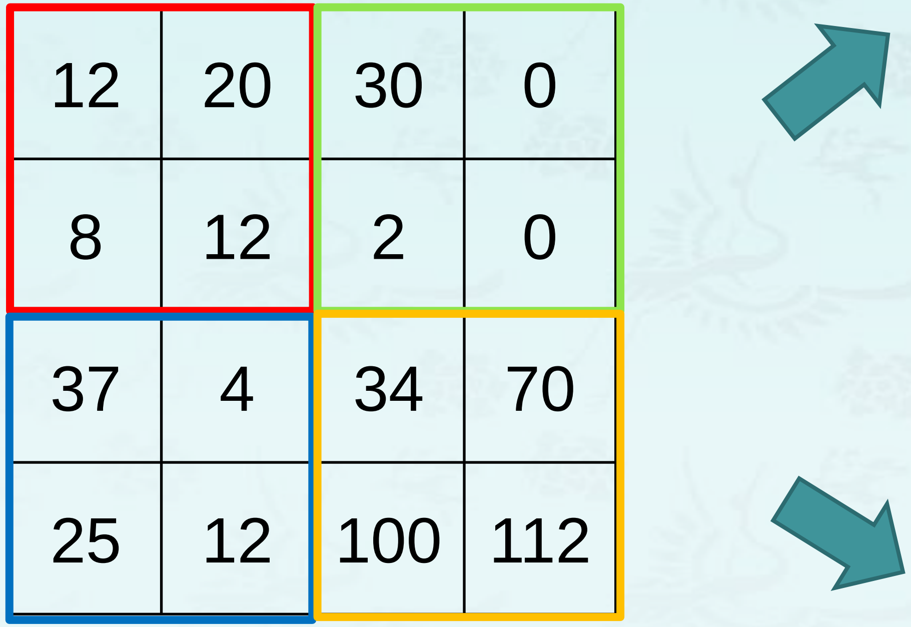
12	20	30	0
8	12	2	0
37	4	34	70
25	12	100	112

- **Max Pooling**

20	30
37	112

- **Average Pooling**

3. CNN 구현: Pooling Layer



12	20	30	0
8	12	2	0
37	4	34	70
25	12	100	112

- **Max Pooling**

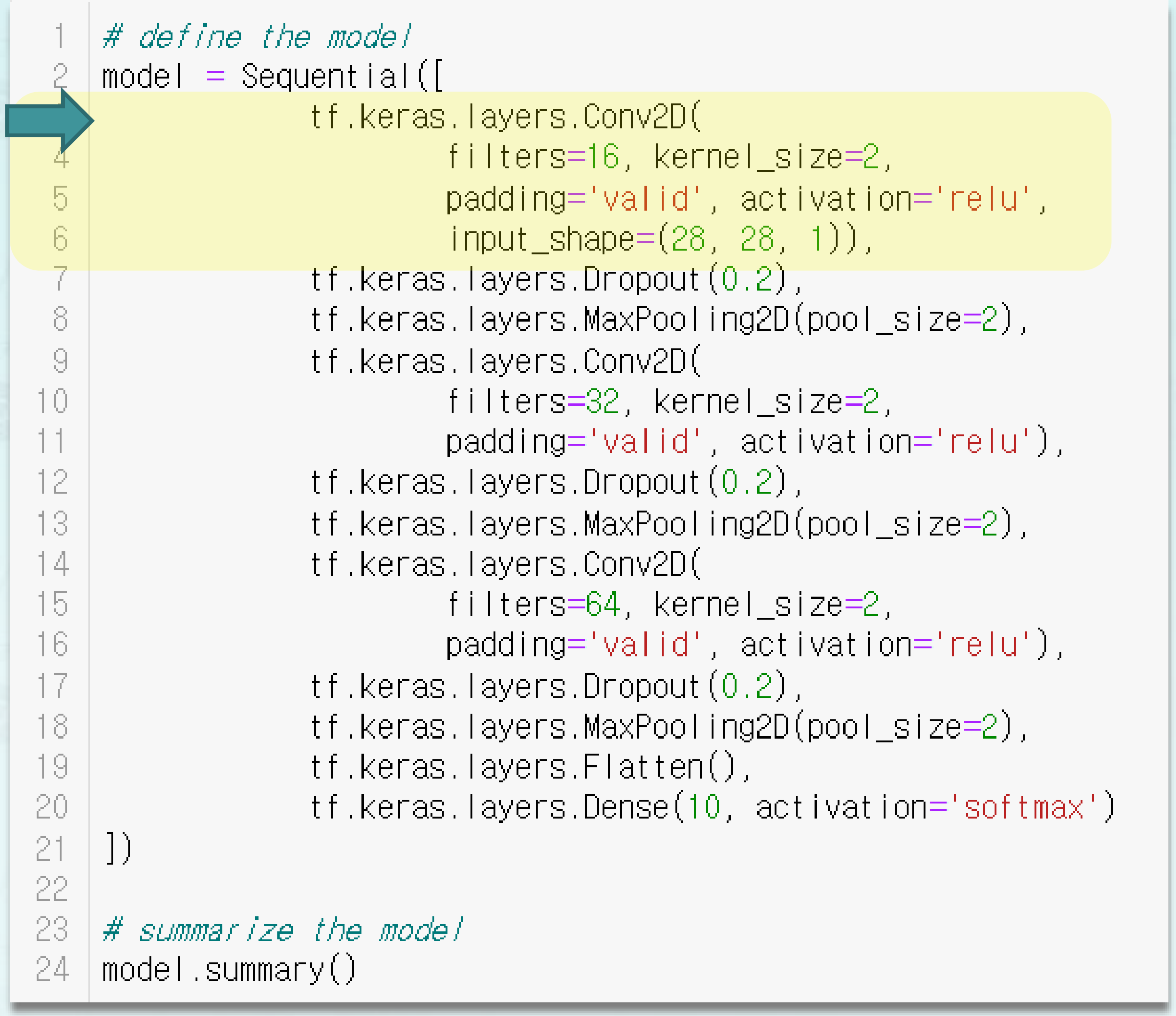
20	30
37	112

- **Average Pooling**

13	8
20	79

3. CNN 구현: MNIST 데이터 분석

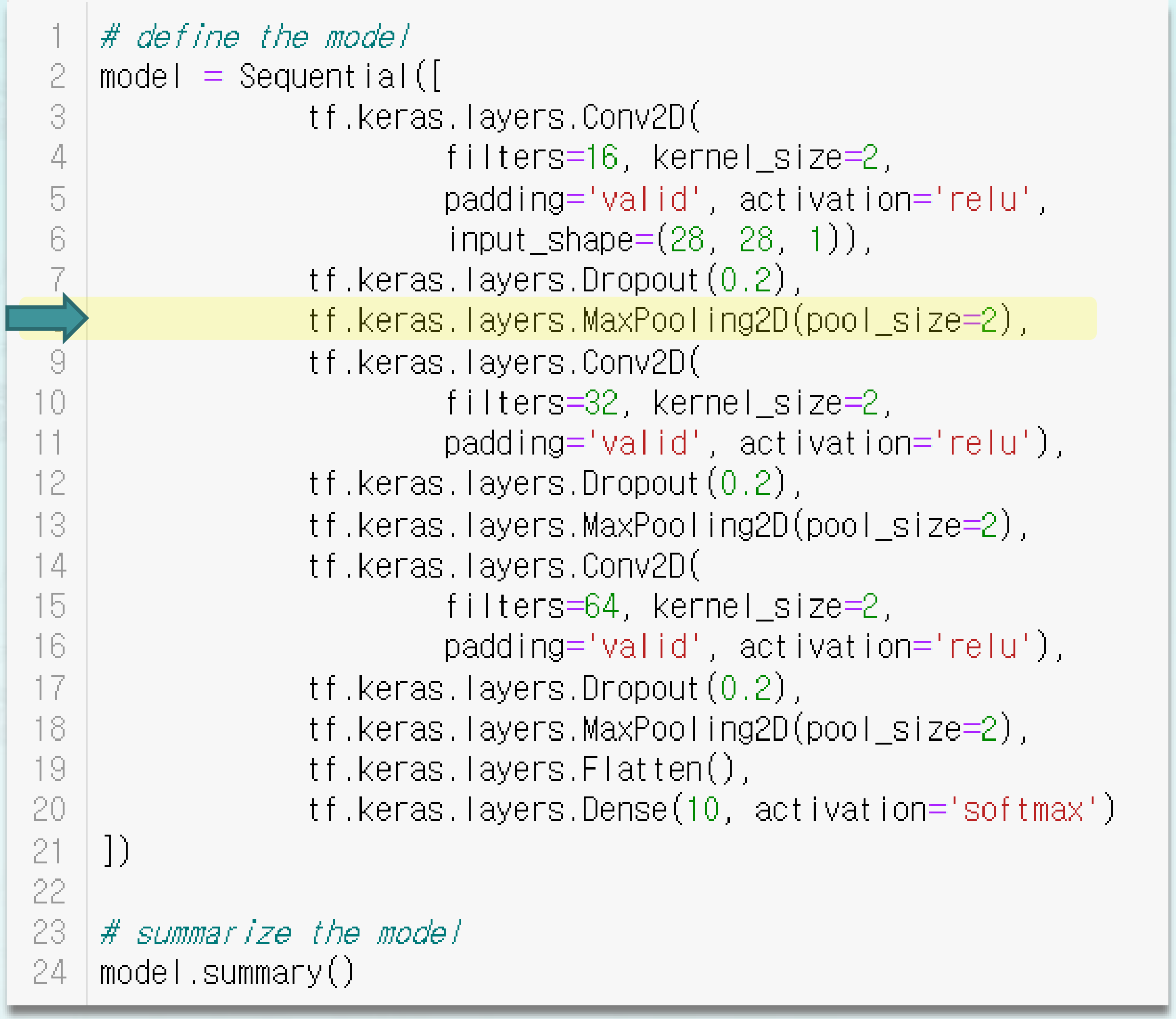
- **kernel_size = 2**
- **filters = 16**



```
# define the model
model = Sequential([
            tf.keras.layers.Conv2D(
                    filters=16, kernel_size=2,
                    padding='valid', activation='relu',
                    input_shape=(28, 28, 1)),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=32, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=64, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(10, activation='softmax')
])

# summarize the model
model.summary()
```

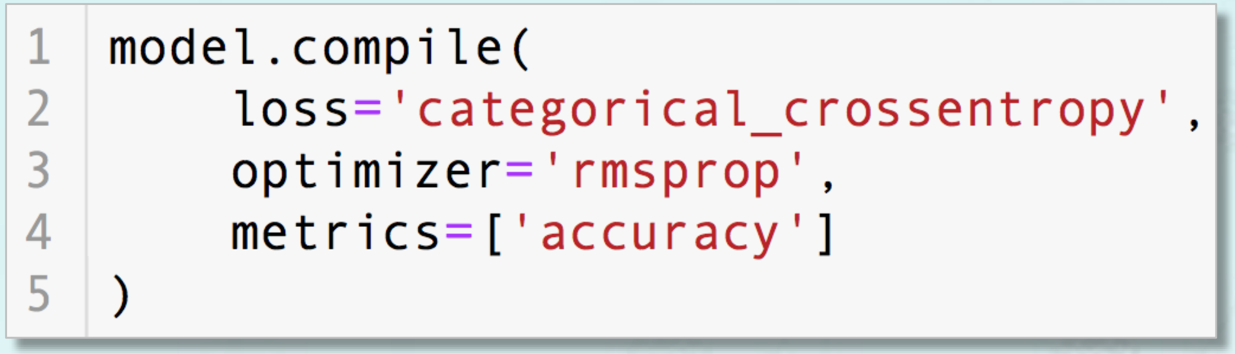
3. CNN 구현: MNIST 데이터 분석

- **MaxPooling2D()**

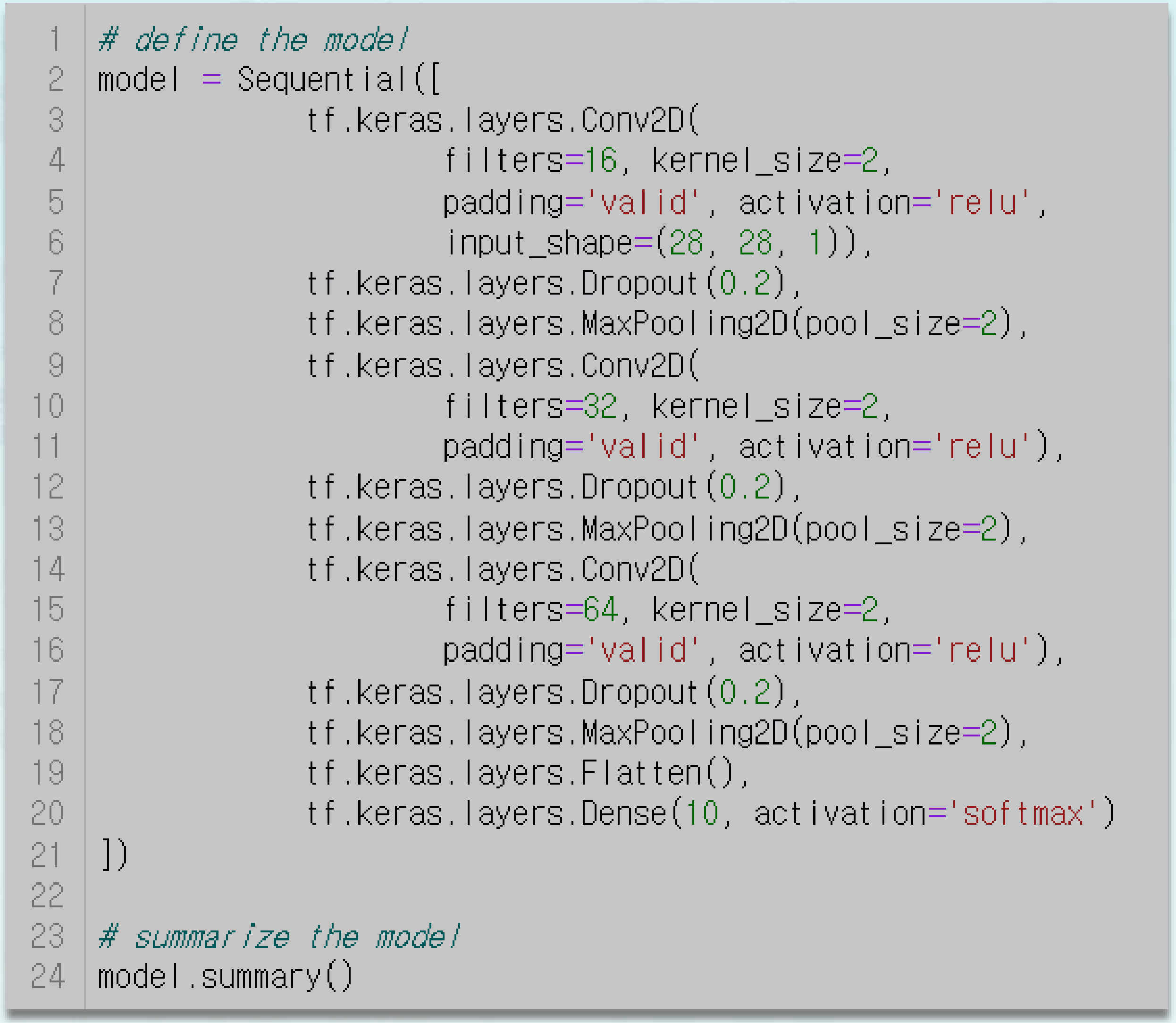


```python
# define the model
model = Sequential([
        tf.keras.layers.Conv2D(
                filters=16, kernel_size=2,
                padding='valid', activation='relu',
                input_shape=(28, 28, 1)),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.MaxPooling2D(pool_size=2),
        tf.keras.layers.Conv2D(
                filters=32, kernel_size=2,
                padding='valid', activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.MaxPooling2D(pool_size=2),
        tf.keras.layers.Conv2D(
                filters=64, kernel_size=2,
                padding='valid', activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.MaxPooling2D(pool_size=2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10, activation='softmax')
])

# summarize the model
model.summary()
```

3. CNN 구현: MNIST 데이터 분석

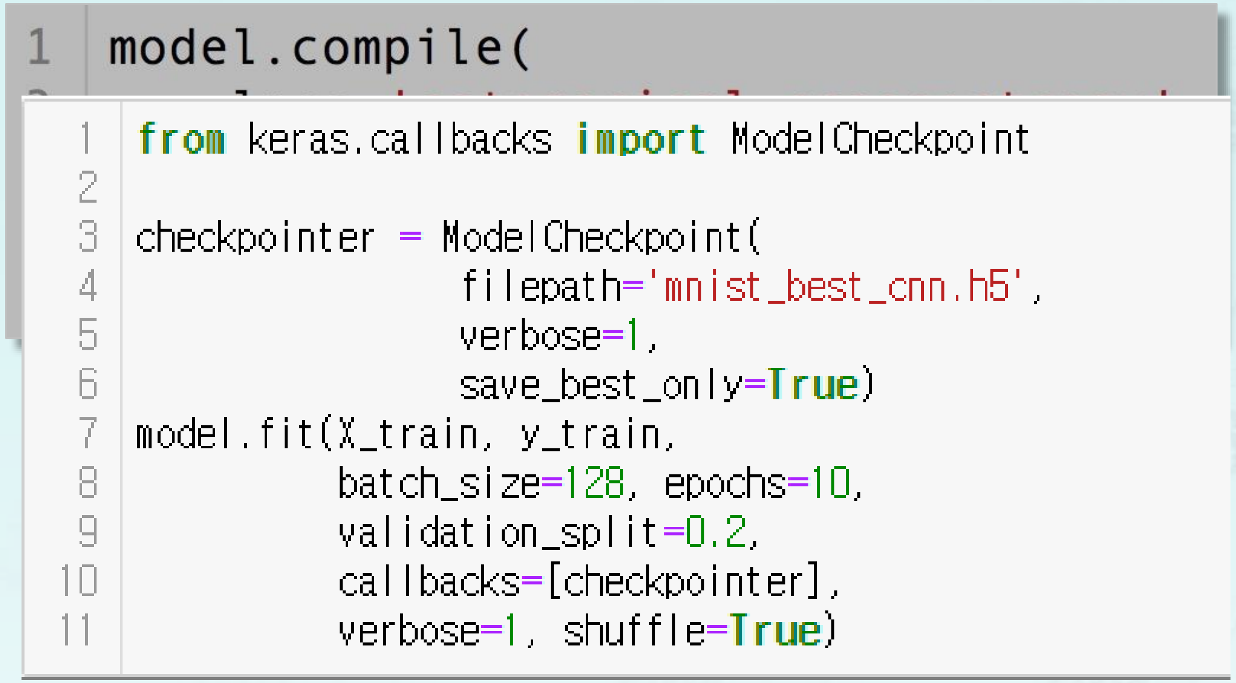


```
model.compile(
    loss='categorical_crossentropy',
    optimizer='rmsprop',
    metrics=['accuracy']
)
```



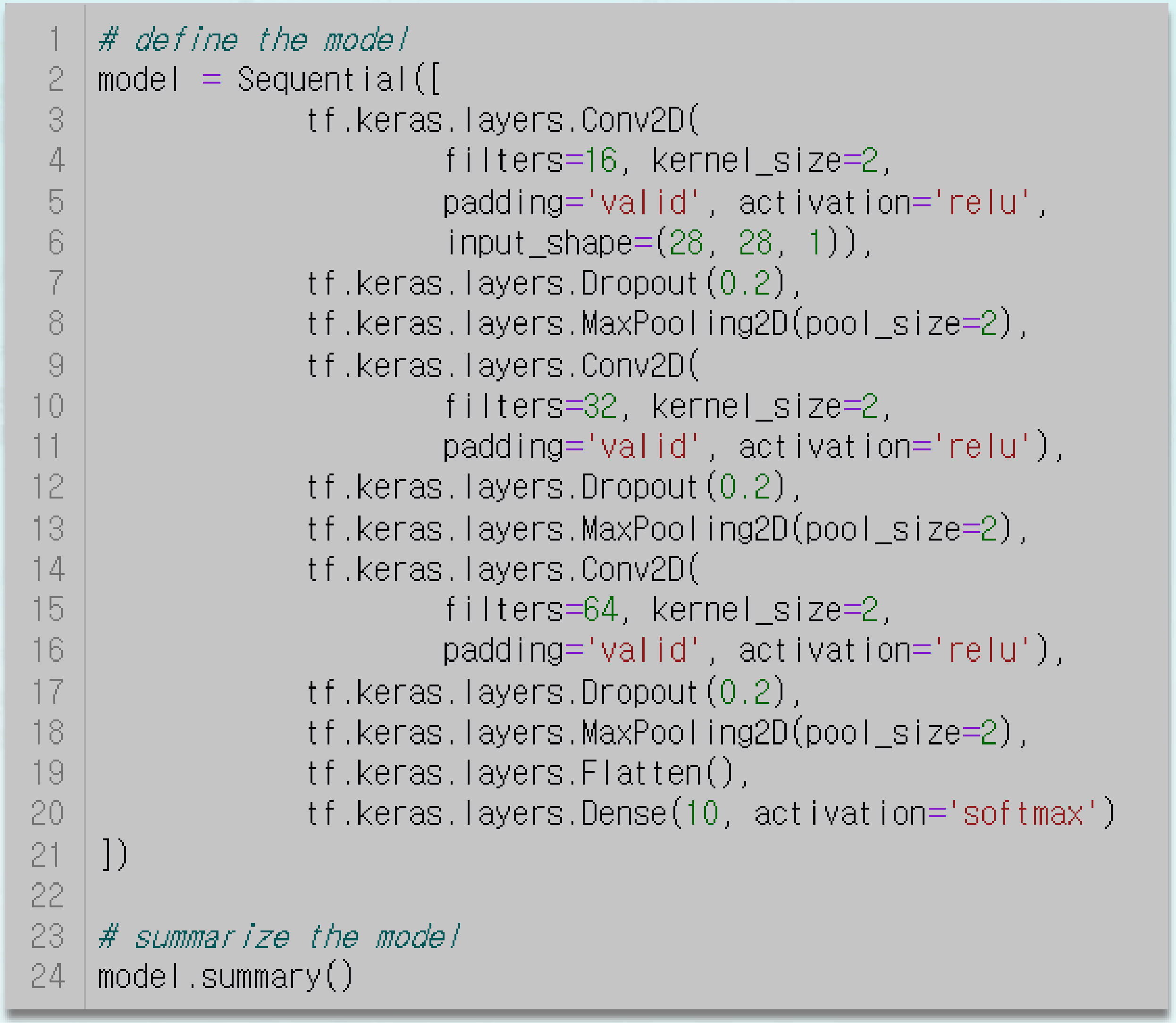
```
# define the model
model = Sequential([
            tf.keras.layers.Conv2D(
                    filters=16, kernel_size=2,
                    padding='valid', activation='relu',
                    input_shape=(28, 28, 1)),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=32, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=64, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(10, activation='softmax')
])

# summarize the model
model.summary()
```

3. CNN 구현: MNIST 데이터 분석



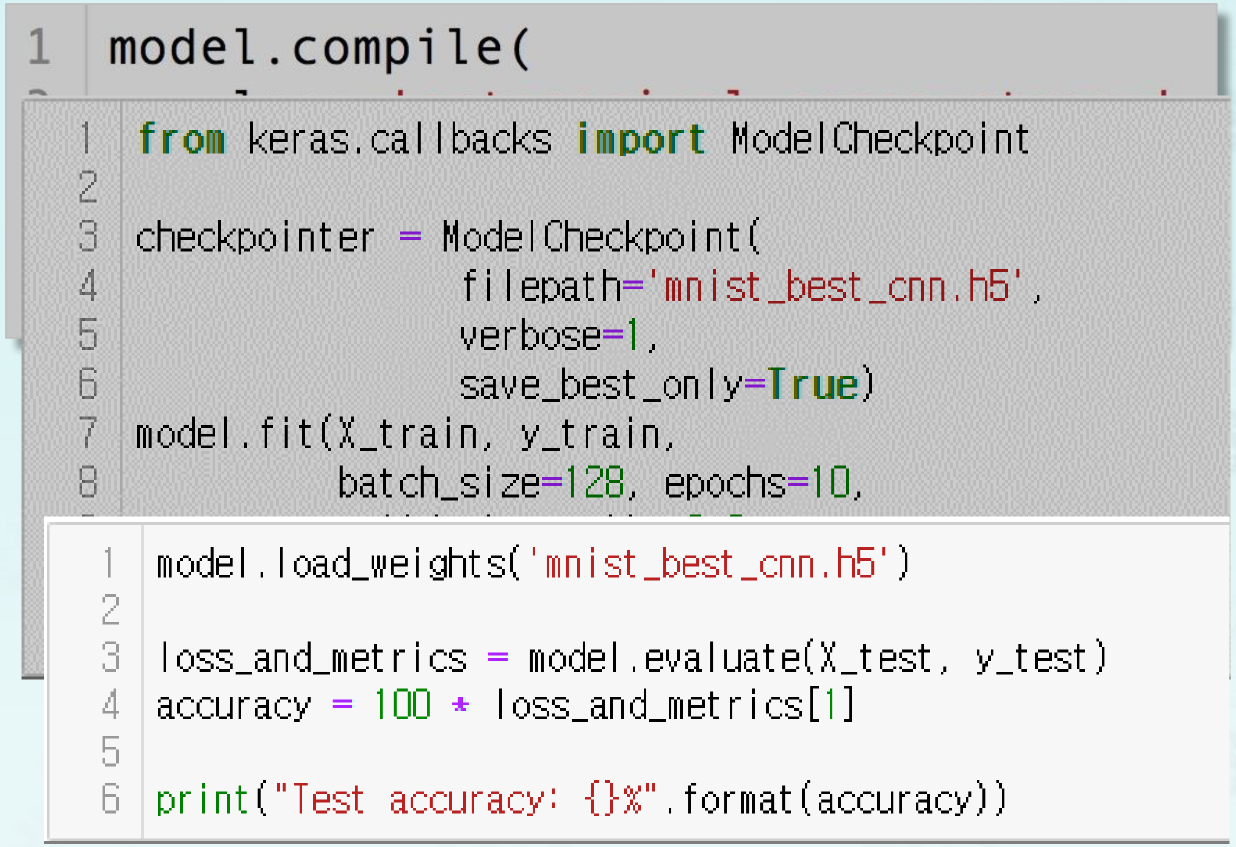
```
model.compile(
```

```
from keras.callbacks import ModelCheckpoint

checkpointer = ModelCheckpoint(
                    filepath='mnist_best_cnn.h5',
                    verbose=1,
                    save_best_only=True)
model.fit(X_train, y_train,
            batch_size=128, epochs=10,
            validation_split=0.2,
            callbacks=[checkpointer],
            verbose=1, shuffle=True)
```



```
# define the model
model = Sequential([
            tf.keras.layers.Conv2D(
                    filters=16, kernel_size=2,
                    padding='valid', activation='relu',
                    input_shape=(28, 28, 1)),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=32, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=64, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(10, activation='softmax')
])

# summarize the model
model.summary()
```

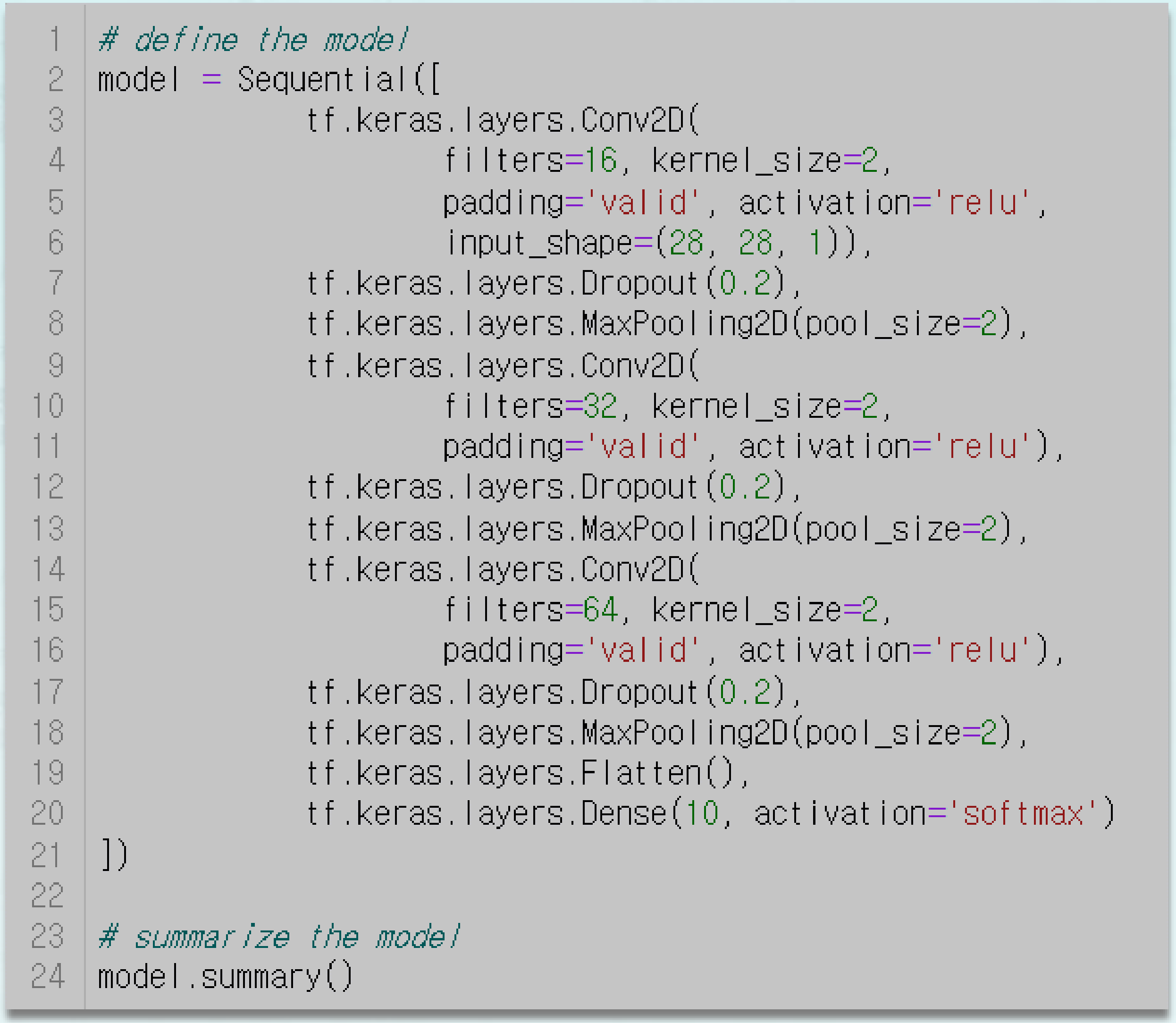
3. CNN 구현: MNIST 데이터 분석



```
model.compile(
```

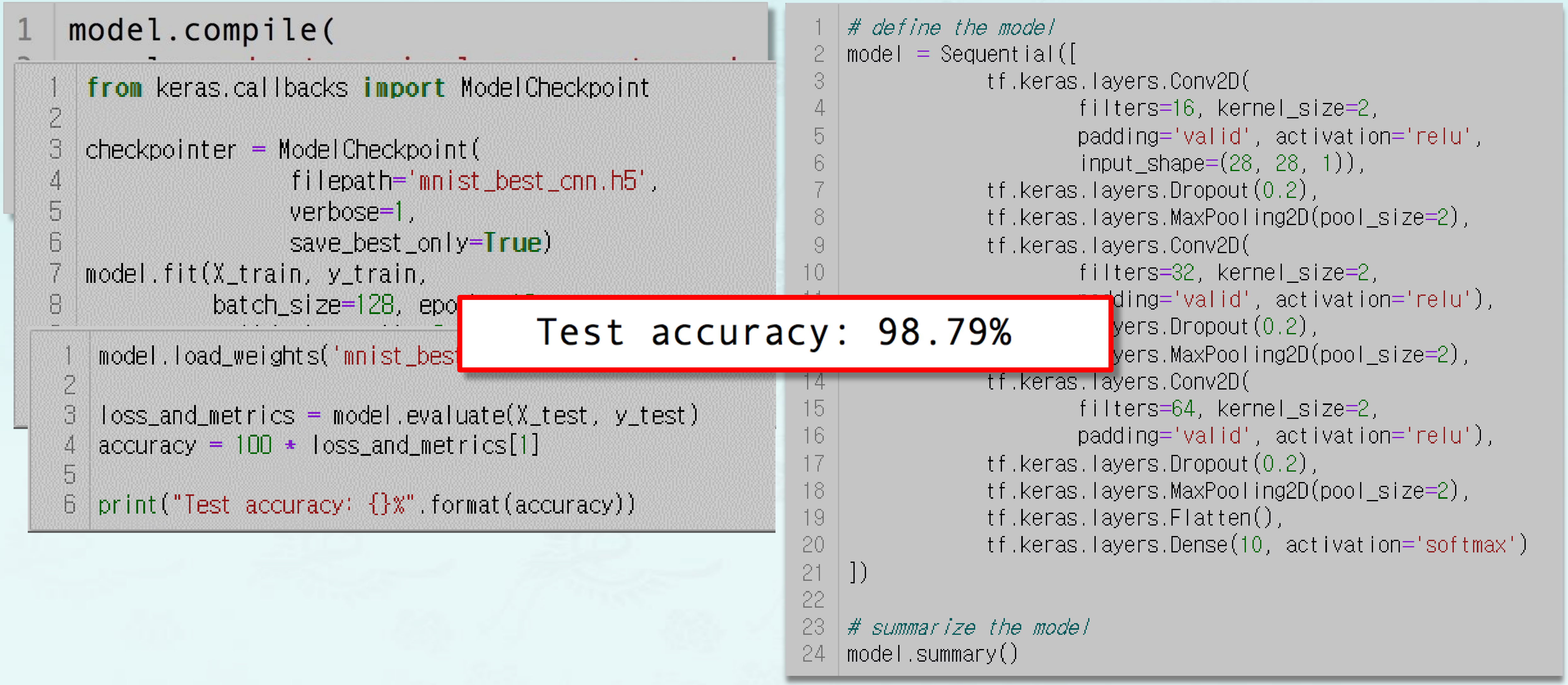
```
from keras.callbacks import ModelCheckpoint

checkpointer = ModelCheckpoint(
                    filepath='mnist_best_cnn.h5',
                    verbose=1,
                    save_best_only=True)
model.fit(X_train, y_train,
            batch_size=128, epochs=10,
```

```
model.load_weights('mnist_best_cnn.h5')

loss_and_metrics = model.evaluate(X_test, y_test)
accuracy = 100 * loss_and_metrics[1]

print("Test accuracy: {}%".format(accuracy))
```



```
# define the model
model = Sequential([
            tf.keras.layers.Conv2D(
                    filters=16, kernel_size=2,
                    padding='valid', activation='relu',
                    input_shape=(28, 28, 1)),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=32, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Conv2D(
                    filters=64, kernel_size=2,
                    padding='valid', activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.MaxPooling2D(pool_size=2),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(10, activation='softmax')
])

# summarize the model
model.summary()
```

3. CNN 구현: MNIST 데이터 분석



```
model.compile(
```

```
from keras.callbacks import ModelCheckpoint

checkpointer = ModelCheckpoint(
                    filepath='mnist_best_cnn.h5',
                    verbose=1,
                    save_best_only=True)
model.fit(X_train, y_train,
            batch_size=128, epo
```

```
model.load_weights('mnist_bes

loss_and_metrics = model.evaluate(X_test, y_test)
accuracy = 100 * loss_and_metrics[1]

print("Test accuracy: {}%".format(accuracy))
```

```
Test accuracy: 98.79%
```

```
# define the model
model = Sequential([
        tf.keras.layers.Conv2D(
                filters=16, kernel_size=2,
                padding='valid', activation='relu',
                input_shape=(28, 28, 1)),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.MaxPooling2D(pool_size=2),
        tf.keras.layers.Conv2D(
                filters=32, kernel_size=2,
                  ding='valid', activation='relu'),
        yers.Dropout(0.2),
        yers.MaxPooling2D(pool_size=2),
        tf.keras.layers.Conv2D(
                filters=64, kernel_size=2,
                padding='valid', activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.MaxPooling2D(pool_size=2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10, activation='softmax')
])

# summarize the model
model.summary()
```

오픈 프레임워크 – 텐서플로우 & 케라스

- 학습 내용
  - 신경망 구현을 위한 오픈 프레임워크는 무엇이 있는지 알아보기.
  - **Tensorflow & Keras**는 무엇인지 이해하기.
  - **Tensorflow & Keras** 를 이용하여 **MNIST** 데이터를 분석하기.
  - **Tensorflow & Keras** 를 이용하여 **CNN**을 구현하기

- 차시 예고
  - 오픈 프레임워크 – **PyTorch**
  - 기계학습 모델 **YOLO**
  - 기계학습 모델 **GAN**

텐서플로우 & 케라스

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수