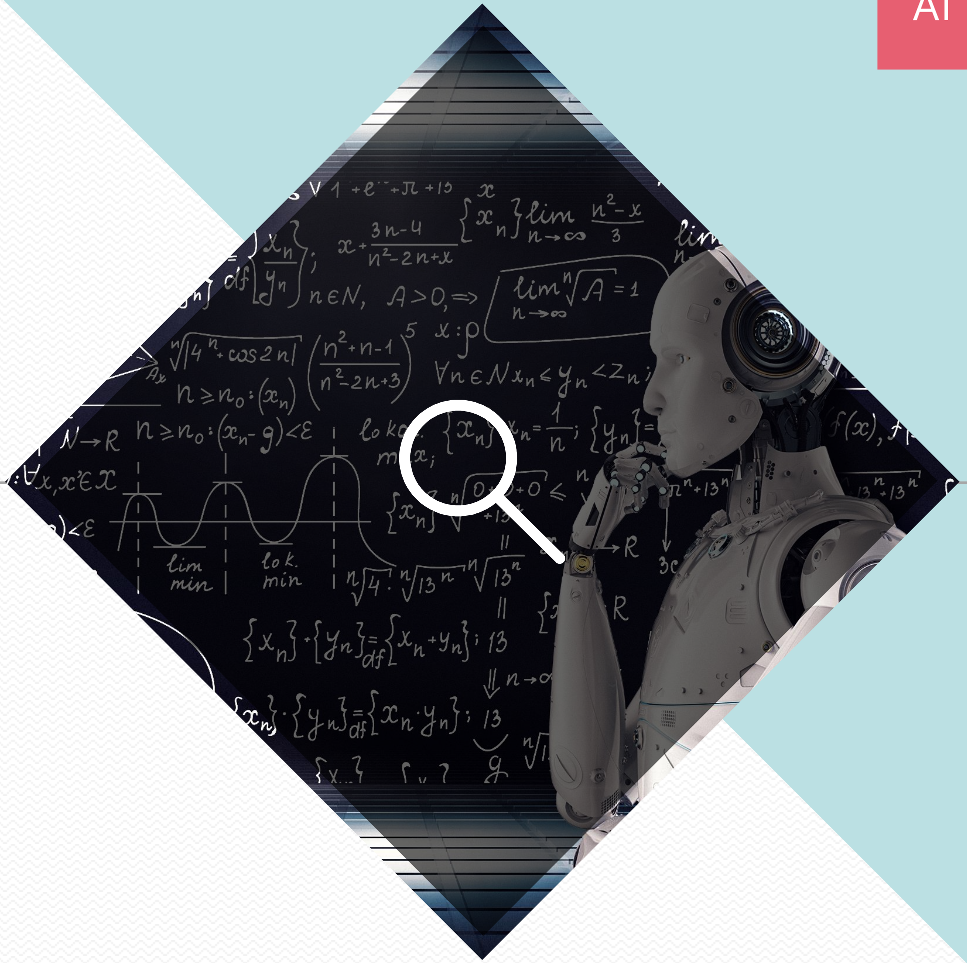
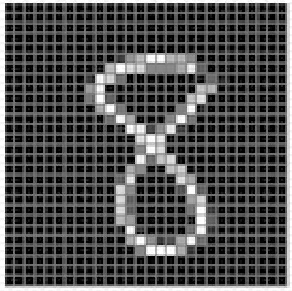
AI TRAINING SERIES



# ANN 실습

FASHION MNIST

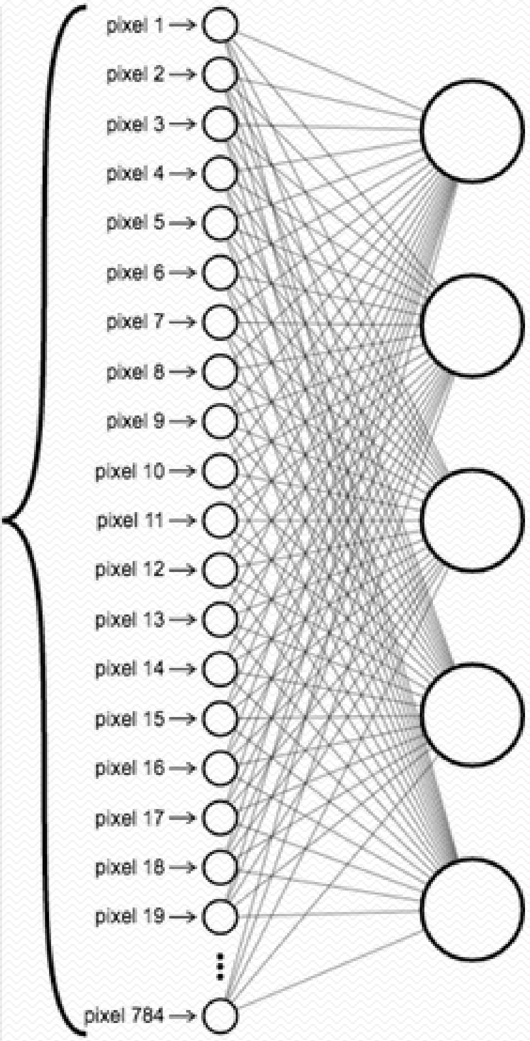
# FEED THE IMAGE DATA



28 x 28
784 pixels

| | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 62 | 146 | 182 | 254 | 254 | 181 | 176 | 139 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 34 | 186 | 253 | 217 | 208 | 136 | 136 | 136 | 166 | 232 | 99 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 61 | 242 | 208 | 111 | 3 | 0 | 0 | 0 | 0 | 0 | 18 | 32 | 107 | 43 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 156 | 242 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 191 | 181 | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 121 | 255 | 98 | 3 | 0 | 0 | 0 | 0 | 0 | 8 | 194 | 225 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 169 | 253 | 120 | 3 | 0 | 0 | 0 | 0 | 128 | 247 | 51 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 111 | 244 | 169 | 19 | 0 | 14 | 131 | 249 | 117 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 59 | 241 | 235 | 72 | 142 | 229 | 66 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 218 | 254 | 231 | 36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 133 | 253 | 221 | 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 237 | 111 | 196 | 217 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 174 | 138 | 0 | 23 | 193 | 204 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 96 | 224 | 0 | 0 | 0 | 25 | 218 | 169 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 215 | 138 | 0 | 0 | 0 | 0 | 86 | 253 | 99 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 215 | 97 | 0 | 0 | 0 | 0 | 3 | 162 | 214 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 215 | 97 | 0 | 0 | 0 | 0 | 0 | 118 | 253 | 68 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 185 | 157 | 0 | 0 | 0 | 0 | 0 | 40 | 254 | 98 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 50 | 244 | 61 | 0 | 0 | 0 | 0 | 112 | 244 | 58 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 174 | 251 | 142 | 59 | 83 | 167 | 244 | 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 133 | 253 | 253 | 253 | 169 | 61 | 3 | 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 |

FLATTEN



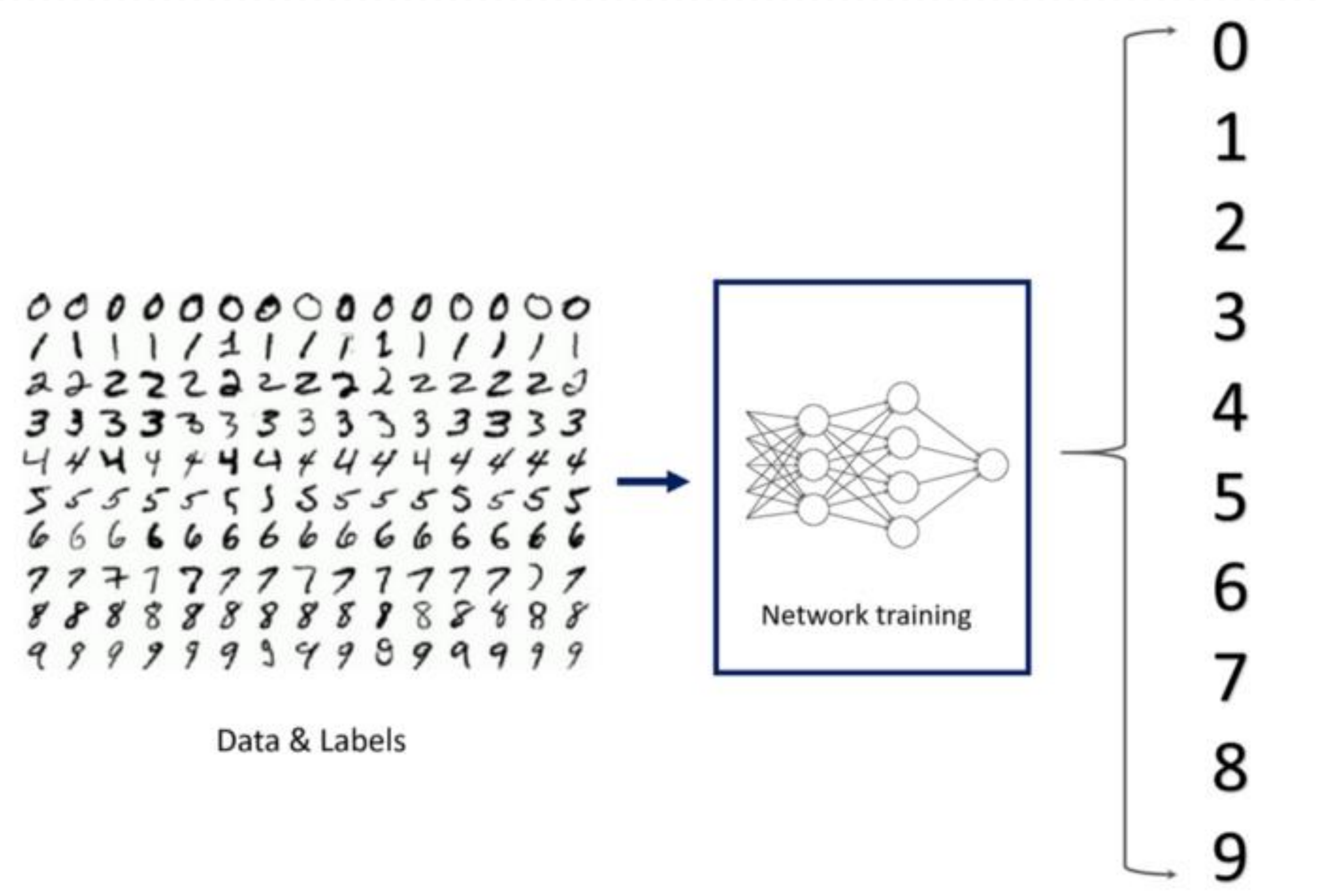
# DATASET PREPARATION

- Dataset
  - **Training dataset:** Data & label that used for training.
  - **Validation dataset:** Data & label that never involved in training.
  - Usually split into **80:20** if dedicated validation set is not given.
- tf.keras.datasets.mnist.load_data()

- Returns Tuple of NumPy arrays: (x_train, y_train), (x_test, y_test)
  - *x_train: uint8 NumPy array of grayscale image data with shapes (60000, 28, 28), containing the training data.*
  - *y_train: uint8 NumPy array of labels (integers in range 0-9) with shape (60000,) for the training data.*
  - *x_test: uint8 NumPy array of grayscale image data with shapes (10000, 28, 28), containing the test data.*
  - *y_test: uint8 NumPy array of labels (integers in range 0-9) with shape (10000,) for the test data.*

# MODEL CONSTRUCTION

- tf.keras.Sequential()
- tf.keras.layers
  - Flatten
  - Dense(ANN/FC)



# MODEL COMPILATION

- tf.keras.Sequential.compile()
  - optimizer
  - loss
  - metrics

# MODEL TRAINING

- tf.model.fit()
  - validation_data
  - epochs
- Hyper parameters
  - Epochs
  - Dataset size
  - Batch size
  - Optimizer / loss function

```
Epoch 1/10
250/250 [==============================] - 210s 840ms/step - loss: 0.6527 -
accuracy: 0.6428 - val_loss: 0.5764 - val_accuracy: 0.6965
Epoch 2/10
250/250 [==============================] - 222s 888ms/step - loss: 0.5223 -
accuracy: 0.7293 - val_loss: 0.5381 - val_accuracy: 0.7275
Epoch 3/10
250/250 [==============================] - 225s 898ms/step - loss: 0.4456 -
accuracy: 0.7894 - val_loss: 0.5187 - val_accuracy: 0.7500
Epoch 4/10
250/250 [==============================] - 224s 895ms/step - loss: 0.3441 -
accuracy: 0.8429 - val_loss: 0.5366 - val_accuracy: 0.7545
Epoch 5/10
250/250 [==============================] - 225s 899ms/step - loss: 0.2362 -
accuracy: 0.9046 - val_loss: 0.6528 - val_accuracy: 0.7575
Epoch 6/10
250/250 [==============================] - 238s 953ms/step - loss: 0.1323 -
accuracy: 0.9465 - val_loss: 0.8491 - val_accuracy: 0.7460
Epoch 7/10
250/250 [==============================] - 232s 927ms/step - loss: 0.0696 -
accuracy: 0.9734 - val_loss: 1.0140 - val_accuracy: 0.7570
Epoch 8/10
250/250 [==============================] - 238s 952ms/step - loss: 0.0591 -
accuracy: 0.9801 - val_loss: 1.0195 - val_accuracy: 0.7415
Epoch 9/10
250/250 [==============================] - 242s 967ms/step - loss: 0.0339 -
accuracy: 0.9893 - val_loss: 1.2181 - val_accuracy: 0.7460
Epoch 10/10
250/250 [==============================] - 248s 991ms/step - loss: 0.0341 -
accuracy: 0.9875 - val_loss: 1.3117 - val_accuracy: 0.7545
```

# SIMPLE ANN EXAMPLE WITH KERAS

```
model = Sequential([
  Flatten(),
  Dense(128, activation='sigmoid'),
  Dense(64, activation='sigmoid'),
  Dense(10, activation='softmax'),
], name=“Simple-ANN")

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

model.summary()
```

```
Model: “Simple-ANN"
___________________________________________
Layer (type) Output Shape Param #
===========================================
flatten (Flatten) (32, 784) 0
___________________________________________
dense (Dense) (32, 128) 100480
___________________________________________
dense_1 (Dense) (32, 64) 8256
___________________________________________
dense_2 (Dense) (32, 10) 650
===========================================
Total params: 109,386
Trainable params: 109,386
Non-trainable params: 0
```

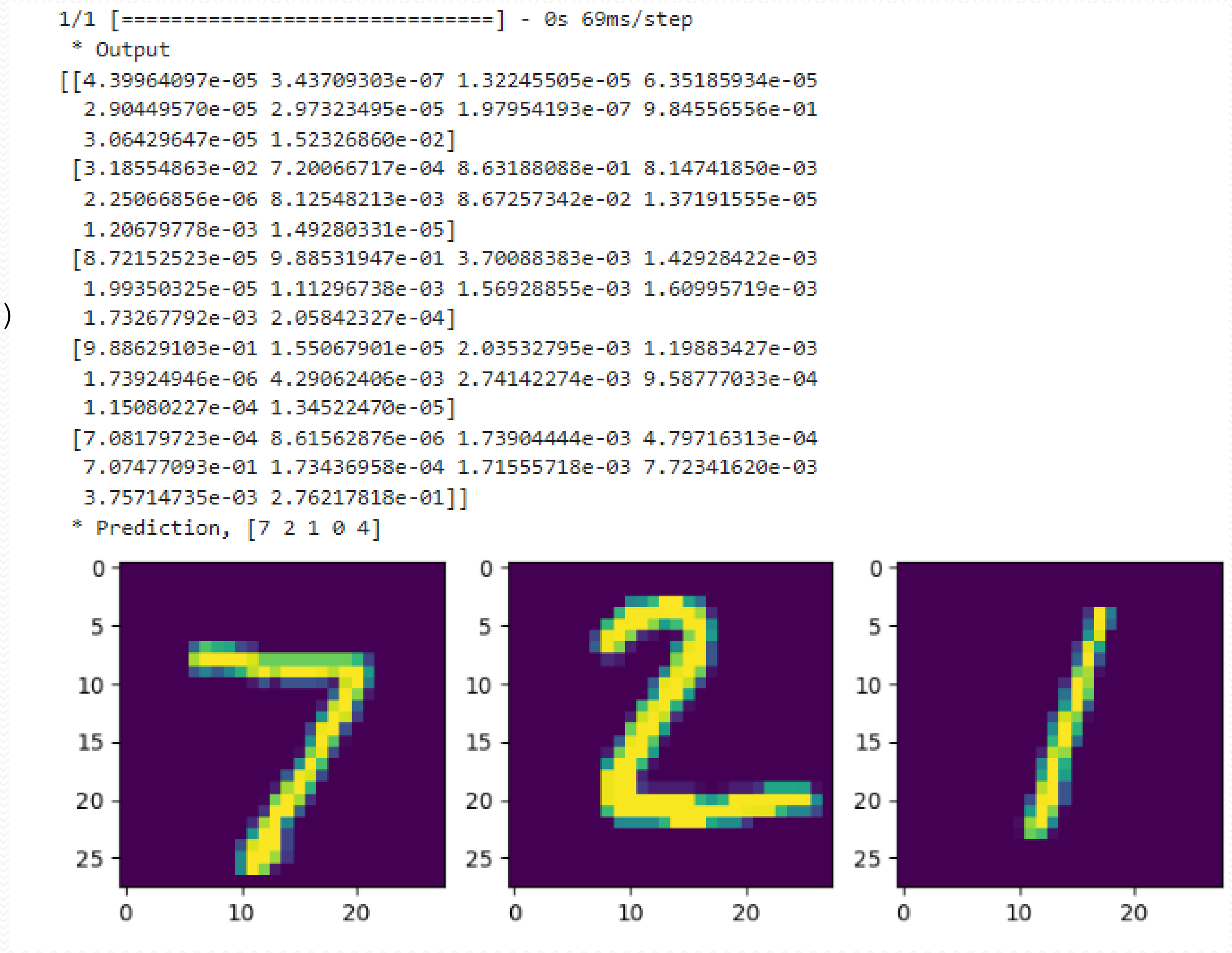
# MODEL INFERENCING

- tf.model.predict()

```
# draw test images with predicted value
NUM = 5
predict = model.predict(image_test[0:NUM])
print(predict)

print(" * Prediction,",
\       np.argmax(predict, axis =1))

plt.figure(figsize=(15,15))
for idx in range(NUM):
  sp = plt.subplot(1,5,idx+1)
  plt.imshow(image_test[idx])
Plt.show()
```



```
1/1 [==============================] - 0s 69ms/step
 * Output
[[4.39964097e-05 3.43709303e-07 1.32245505e-05 6.35185934e-05
  2.90449570e-05 2.97323495e-05 1.97954193e-07 9.84556556e-01
  3.06429647e-05 1.52326860e-02]
 [3.18554863e-02 7.20066717e-04 8.63188088e-01 8.14741850e-03
  2.25066856e-06 8.12548213e-03 8.67257342e-02 1.37191555e-05
  1.20679778e-03 1.49280331e-05]
 [8.72152523e-05 9.88531947e-01 3.70088383e-03 1.42928422e-03
  1.99350325e-05 1.11296738e-03 1.56928855e-03 1.60995719e-03
  1.73267792e-03 2.05842327e-04]
 [9.88629103e-01 1.55067901e-05 2.03532795e-03 1.19883427e-03
  1.73924946e-06 4.29062406e-03 2.74142274e-03 9.58777033e-04
  1.15080227e-04 1.34522470e-05]
 [7.08179723e-04 8.61562876e-06 1.73904444e-03 4.79716313e-04
  7.07477093e-01 1.73436958e-04 1.71555718e-03 7.72341620e-03
  3.75714735e-03 2.76217818e-01]]
 * Prediction, [7 2 1 0 4]
```

# DATASET PREPARATION

- Dataset
  - **Training dataset:** Data & label that used for training.
  - **Validation dataset:** Data & label that never involved in training.
  - Usually split into **80:20** if dedicated validation set is not given.
- tf.keras.datasets.fashion_mnist.load_data()

- Returns Tuple of NumPy arrays: (x_train, y_train), (x_test, y_test)
  - *x_train: uint8 NumPy array of grayscale image data with shapes (60000, 28, 28), containing the training data.*
  - *y_train: uint8 NumPy array of labels (integers in range 0-9) with shape (60000,) for the training data.*
  - *x_test: uint8 NumPy array of grayscale image data with shapes (10000, 28, 28), containing the test data.*
  - *y_test: uint8 NumPy array of labels (integers in range 0-9) with shape (10000,) for the test data.*

# **THANK** YOU

# Hands-on Colab

- [Hands-on] MNIST Dataset
  - https://colab.research.google.com/drive/122uCWPt1eR7yrasyG96ak0vXKKQGqa_Y?usp=sharing
- [Hands-on] Coco Dataset
  - https://colab.research.google.com/drive/1CrOUX7Ta-phwMI_Ngj1Ay9_DtyJUezLK?usp=sharing
- [Hands-on] ANN MNIST
  - https://colab.research.google.com/drive/1_ZhB7hwtYCEtfHTwkUthEsLHx3rlehjz?usp=sharing
- [Hands-on]Benchmark app
  - https://colab.research.google.com/drive/1mF99L-U5NJ0KYjf_VY2ZkA6vhcpX7CSh?usp=sharing

# SIMPLE ANN EXAMPLE WITH KERAS

```
# draw test images with predicted value
NUM = 5
predict = model.predict(image_test[NUM])

print(" * Prediction,",np.argmax(predict, axis =1))
plt.imshow(image_test[idx])
```

# SIMPLE ANN EXAMPLE WITH KERAS

```
model = tf.keras.models.Sequential()

model.add(tf.keras.Input(shape=(28,28)))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(128,
activation='tanh’))

model.add(tf.keras.layers.Dense(64,
activation='tanh'))

model.add(tf.keras.layers.Dense(10,
activation="softmax"))


model.compile(

  optimizer='adam',

  loss='categorical_crossentropy',

  metrics=['accuracy'],

)


model.summary()
```

```
Model: “Simple-ANN"

___________________________________________
Layer (type) Output Shape Param #
===========================================
flatten (Flatten) (32, 784) 0

___________________________________________
dense (Dense) (32, 128) 100480

___________________________________________
dense_1 (Dense) (32, 64) 8256

___________________________________________
dense_2 (Dense) (32, 10) 650
===========================================
Total params: 109,386
Trainable params: 109,386
Non-trainable params: 0
```

# DATASET PREPARATION

- Dataset
  - **Training dataset:** Data & label that used for training.
  - **Validation dataset:** Data & label that never involved in training.
  - Usually split into **80:20** if dedicated validation set is not given.
- tf.keras.datasets.fashion_mnist.load_data()

- Returns Tuple of NumPy arrays: (x_train, y_train), (x_test, y_test)
  - *x_train: uint8 NumPy array of grayscale image data with shapes (60000, 28, 28), containing the training data.*
  - *y_train: uint8 NumPy array of labels (integers in range 0-9) with shape (60000,) for the training data.*
  - *x_test: uint8 NumPy array of grayscale image data with shapes (10000, 28, 28), containing the test data.*
  - *y_test: uint8 NumPy array of labels (integers in range 0-9) with shape (10000,) for the test data.*