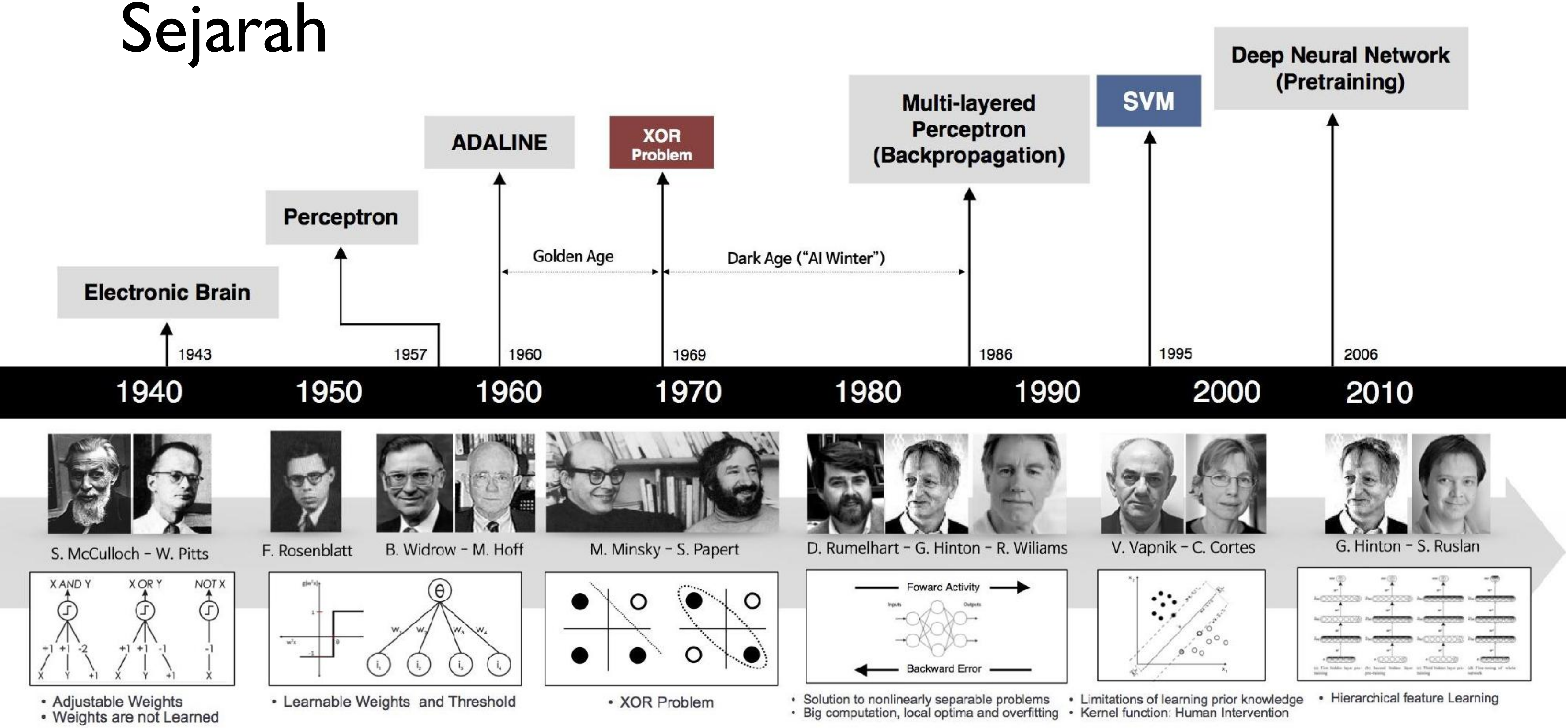
Artificial Neural Network

Made Satria Wibawa, M.Eng.

2020

WHAT & WHY

Sejarah



Electronic Brain — 1943

Perceptron — 1957

ADALINE — 1960

XOR Problem — 1969

Multi-layered Perceptron (Backpropagation) — 1986

SVM — 1995

Deep Neural Network (Pretraining) — 2006

Golden Age

Dark Age ("AI Winter")

1940 1950 1960 1970 1980 1990 2000 2010

S. McCulloch - W. Pitts

- Adjustable Weights
- Weights are not Learned

F. Rosenblatt

B. Widrow - M. Hoff

- Learnable Weights and Threshold

M. Minsky - S. Papert

- XOR Problem

D. Rumelhart - G. Hinton - R. Wiliams

- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting

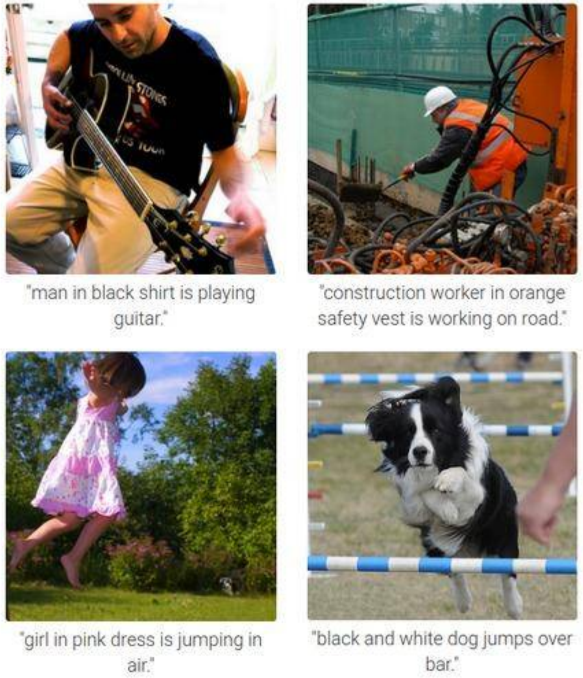
V. Vapnik - C. Cortes

- Limitations of learning prior knowledge
- Kernel function: Human Intervention

G. Hinton - S. Ruslan

- Hierarchical feature Learning

Penerapan



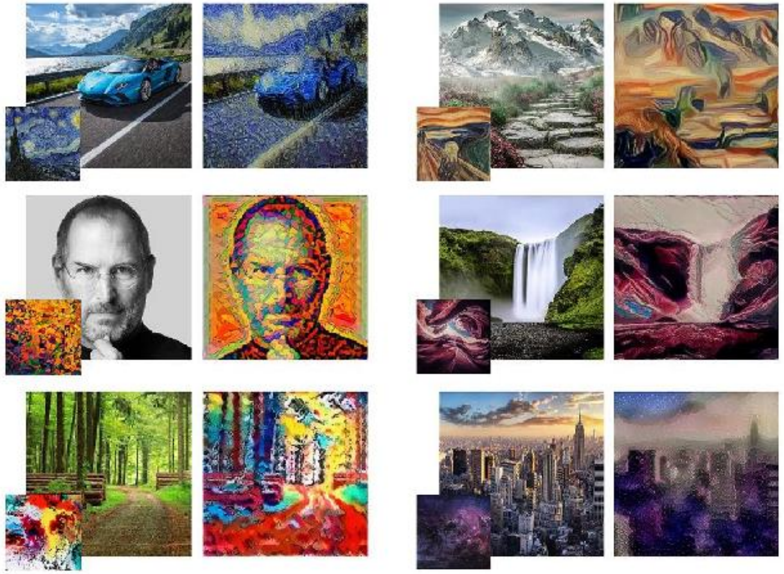
"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

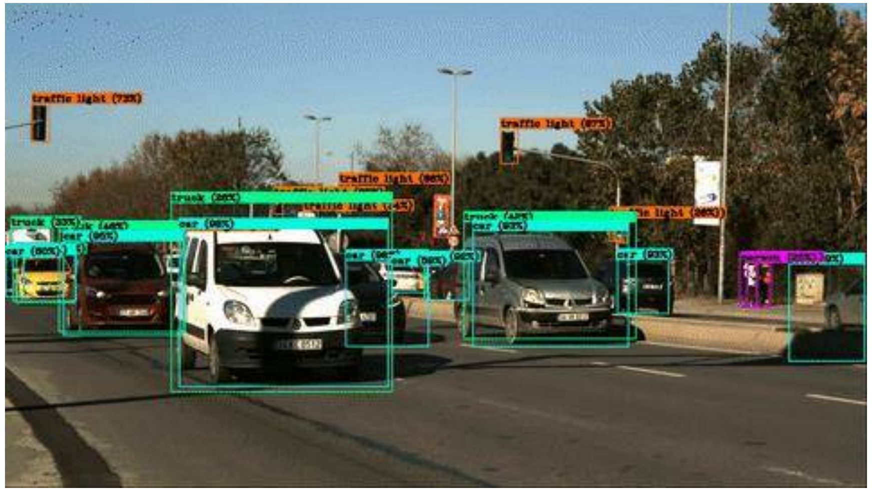
"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

Image Captioning



Style Transfer

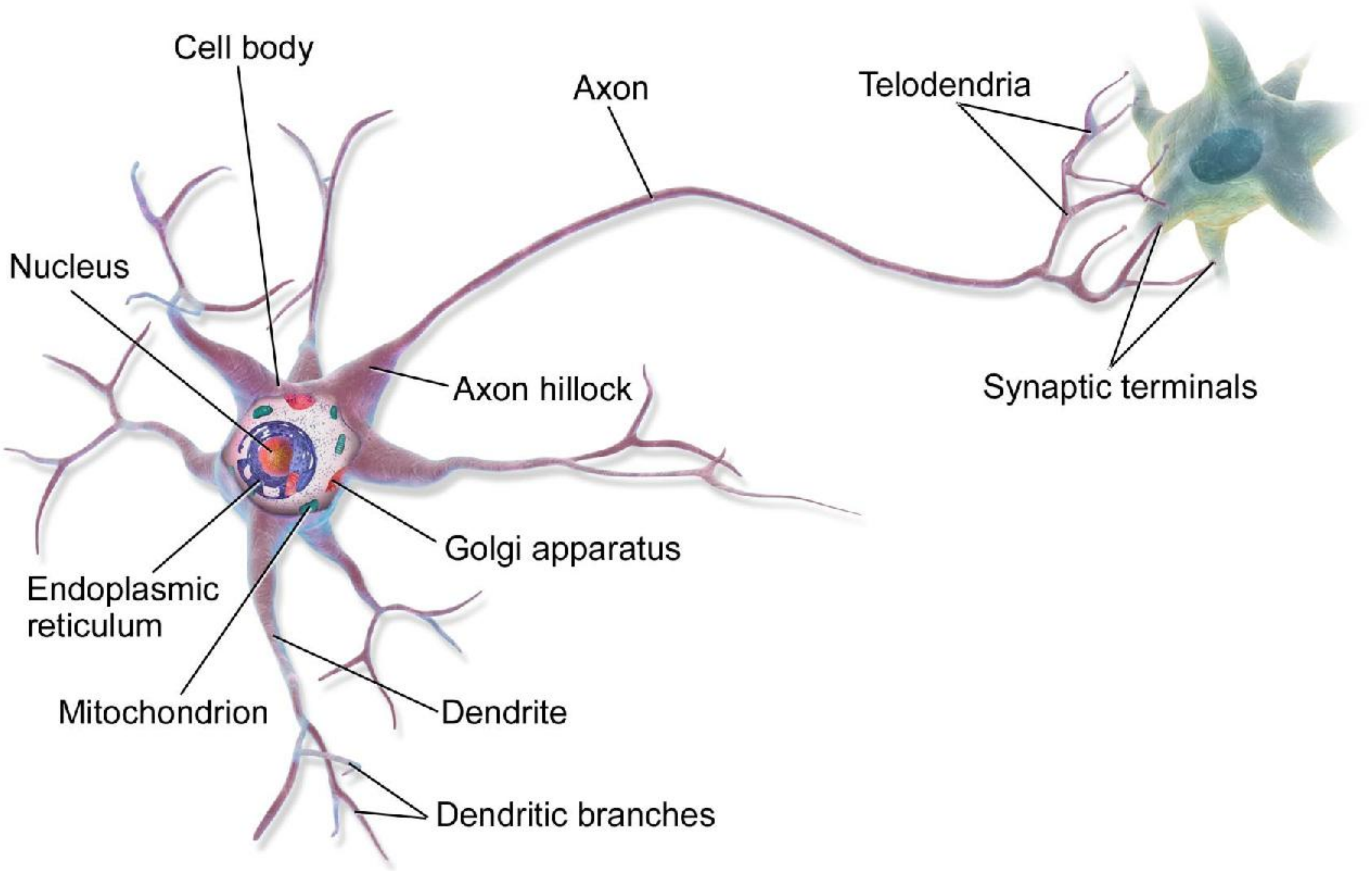


Object Detection

Image Generation

PERCEPTRON

Neuron

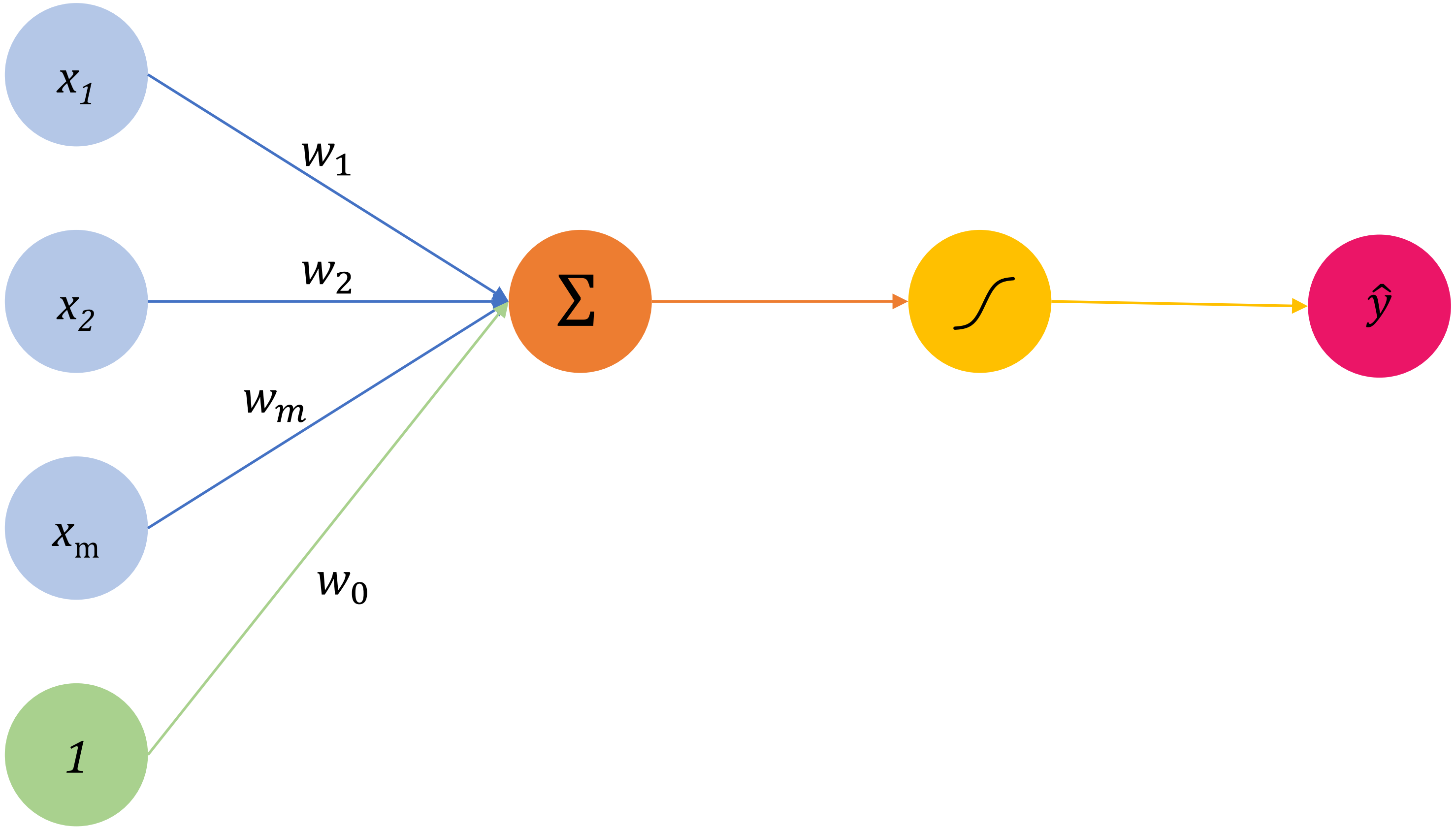


Elemen penting:

1. Dendrite
   menerima sinyal listrik
2. Nucleus
   memproses sinyal listrik dari dendrite
3. Axon
   menghantarkan sinyal listrik dari soma/nucleus
4. Synaptic
   jembatan sinyal listrik ke neuron lainnya

Neuron hanya akan menghantarkan sinyal jika semua sinyal yang diterima dari dendrite cukup kuat

Perceptron: Artificial Neuron



Linear combination of inputs

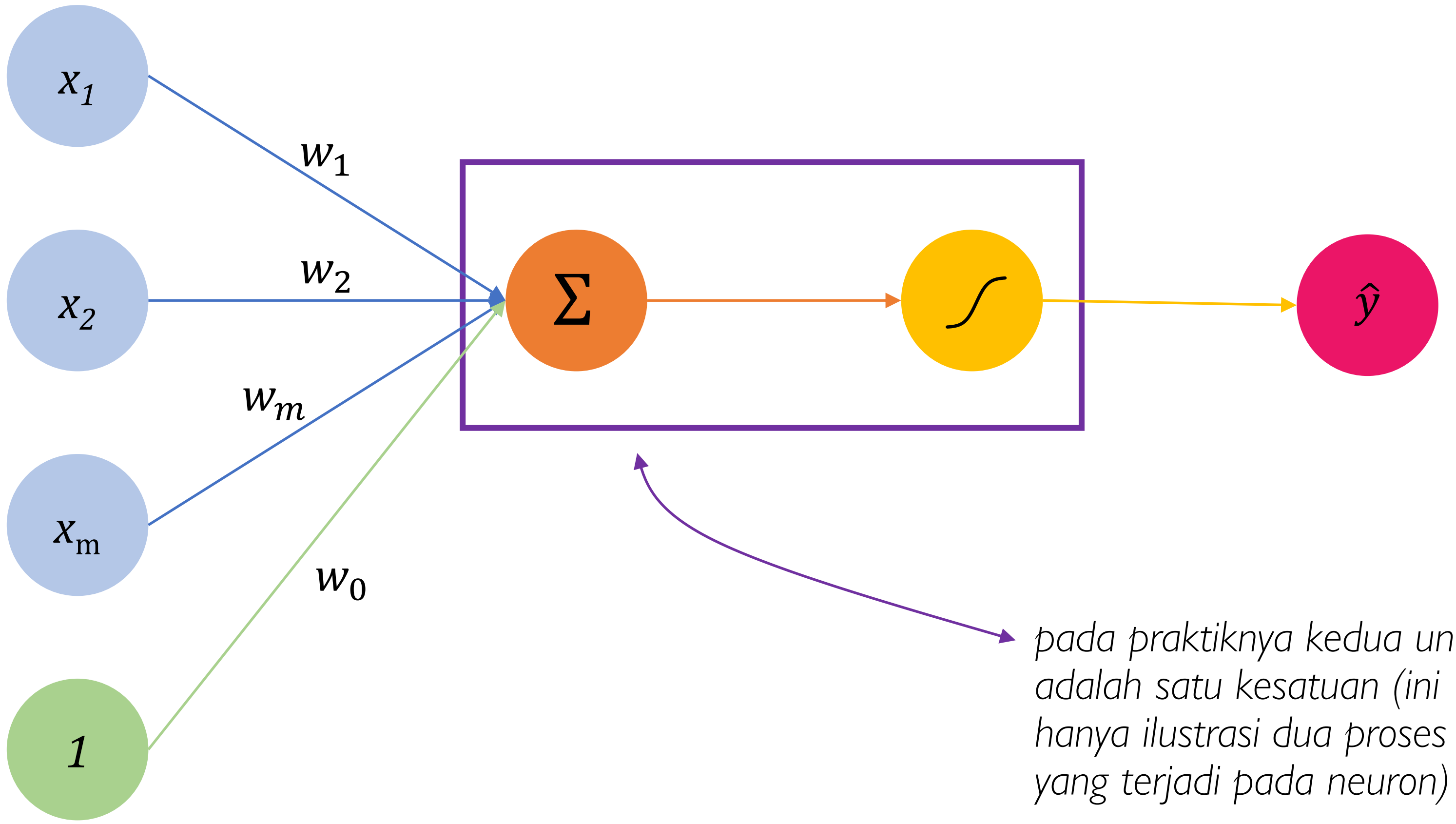
Output

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right)$$

Non-linear activation function

Bias

Perceptron: Artificial Neuron



pada praktiknya kedua unit adalah satu kesatuan (ini hanya ilustrasi dua proses yang terjadi pada neuron)

Linear combination of inputs

Output

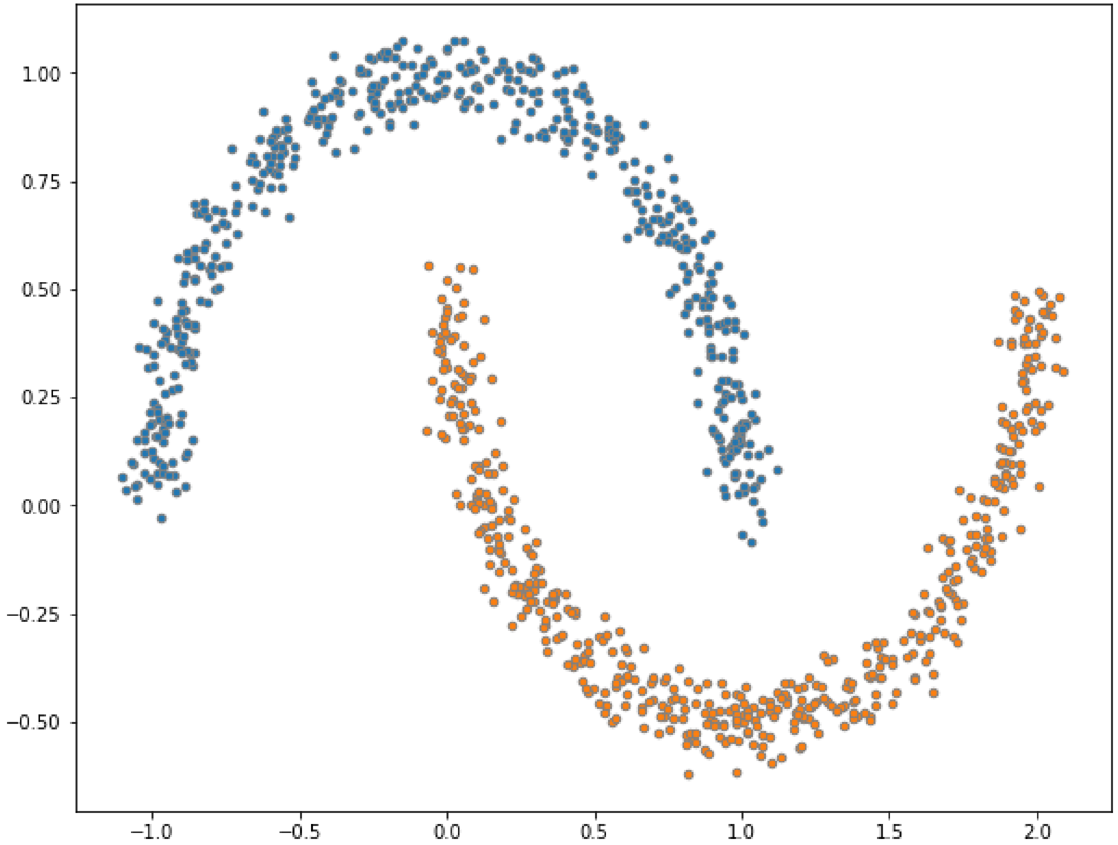
$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right)$$

Non-linear activation function

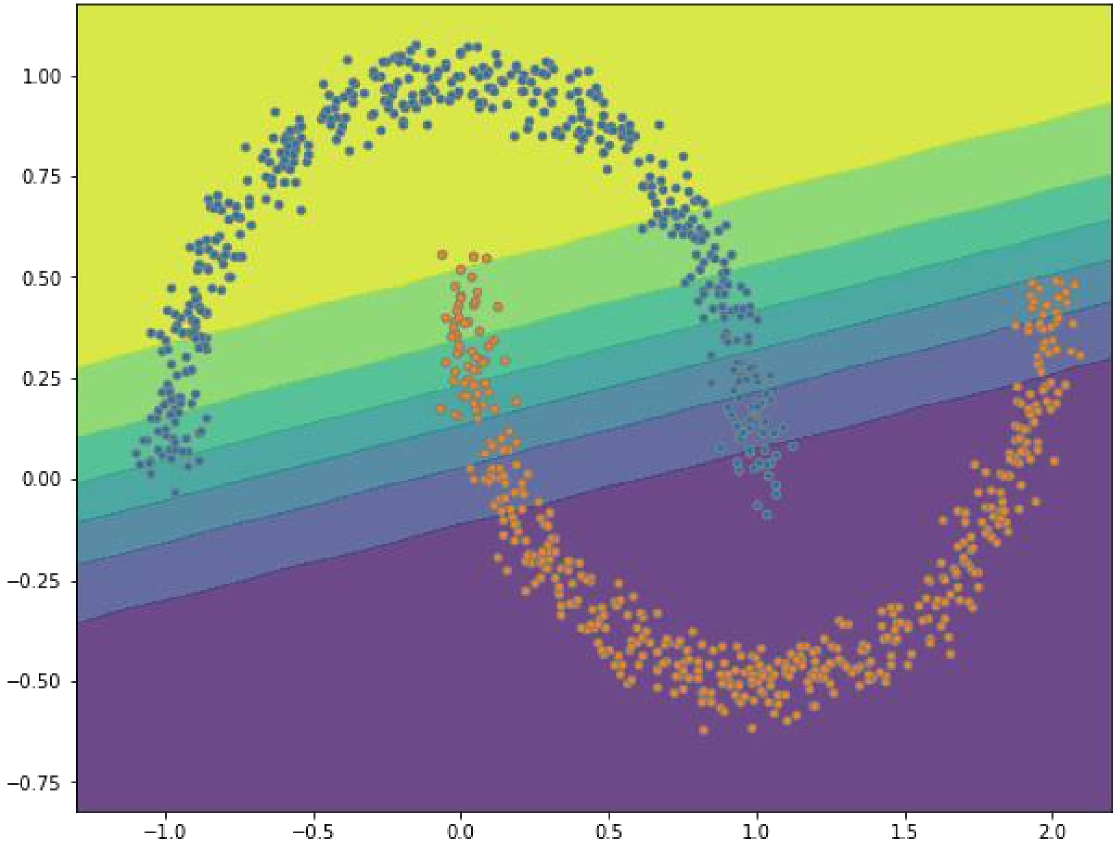
Bias

NON-LINEARITY

Non-Linear Problem

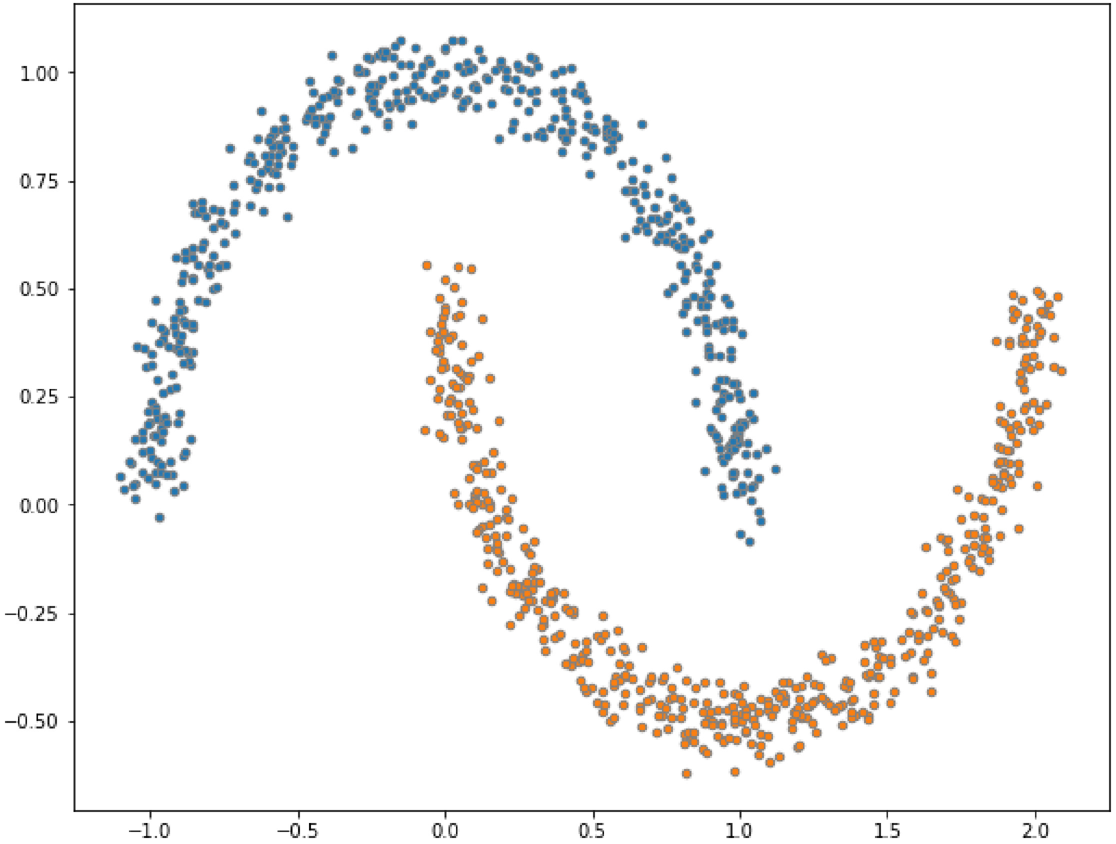


Non-linear Dataset

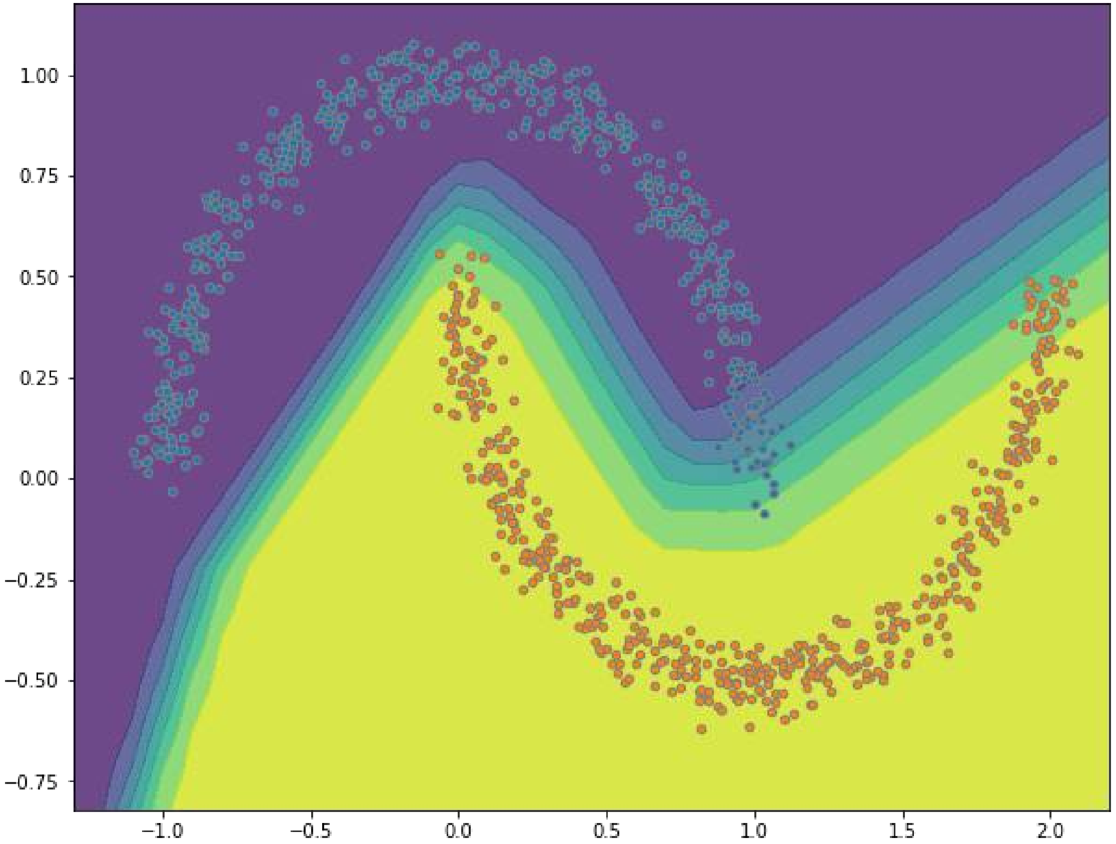


Linear Model (Logistic Regression)

Activation Function: Non-Linearity



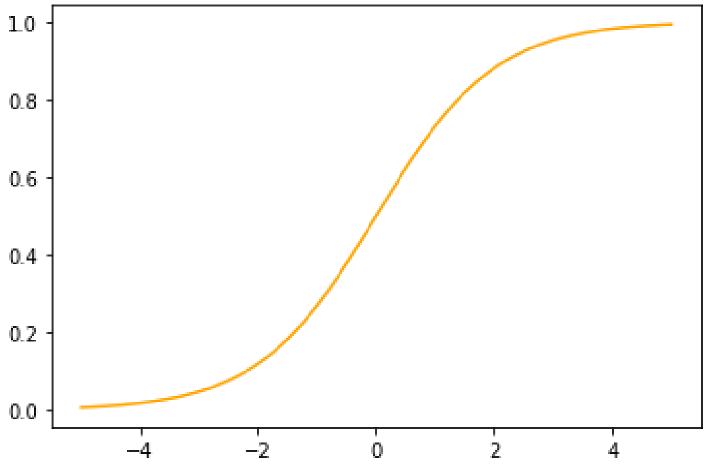
Non-linear Dataset



Non-Linear Model (ANN)

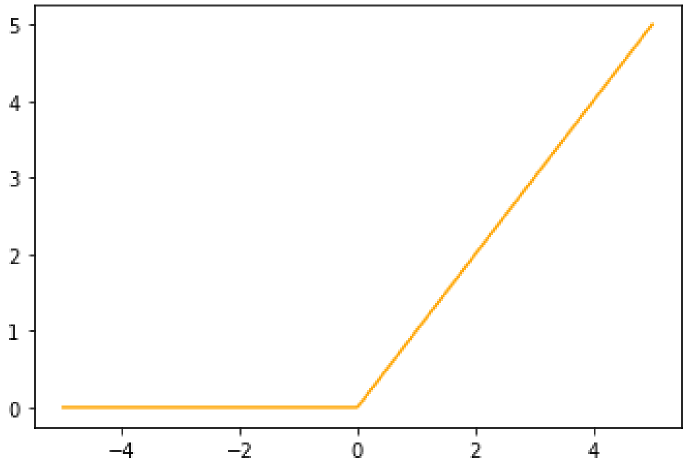
Perceptron: Activation Function

Sigmoid (x)



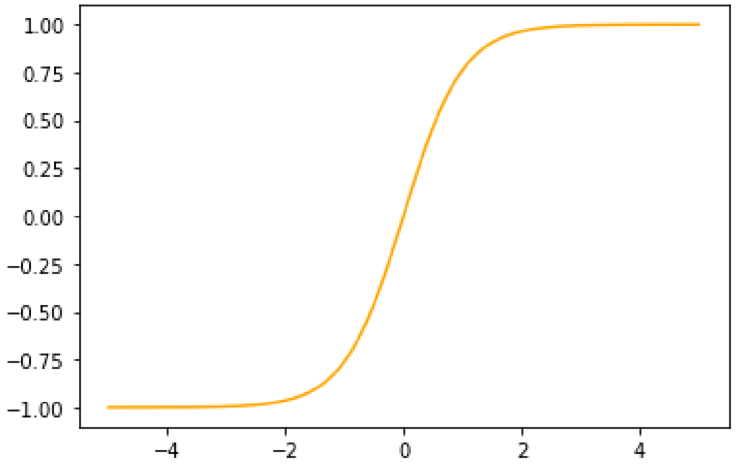
$$g(z) = \frac{1}{1 + e^{-z}}$$

ReLU (x)



$$g(z) = \max(0, x)$$

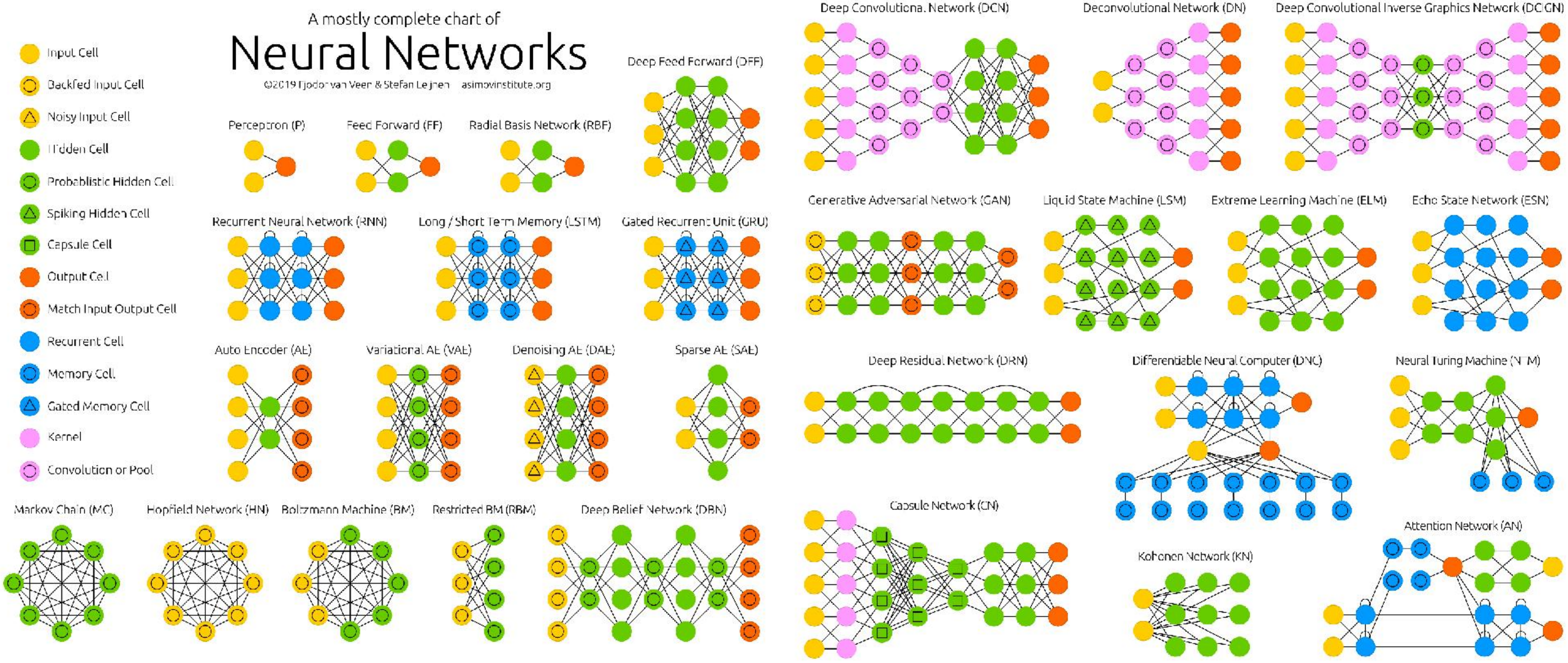
Tanh (x)



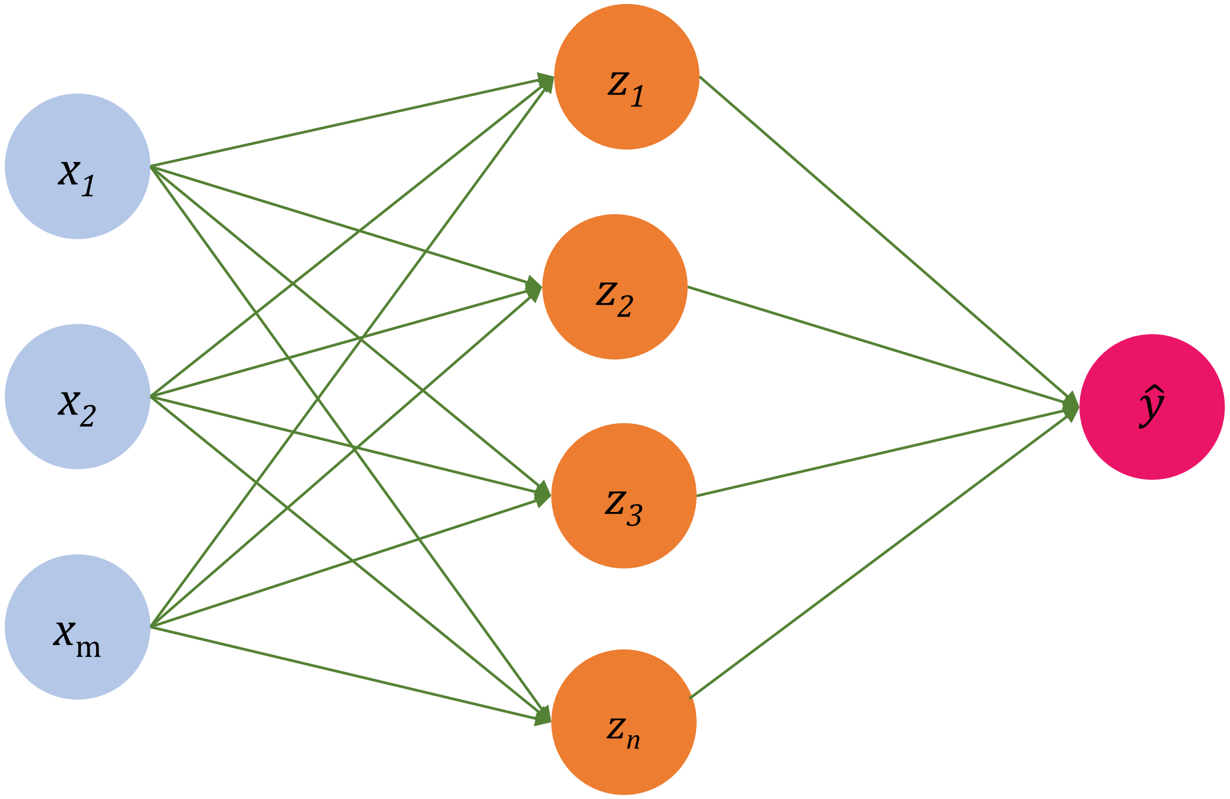
$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

NETWORK ARCHITECTURE

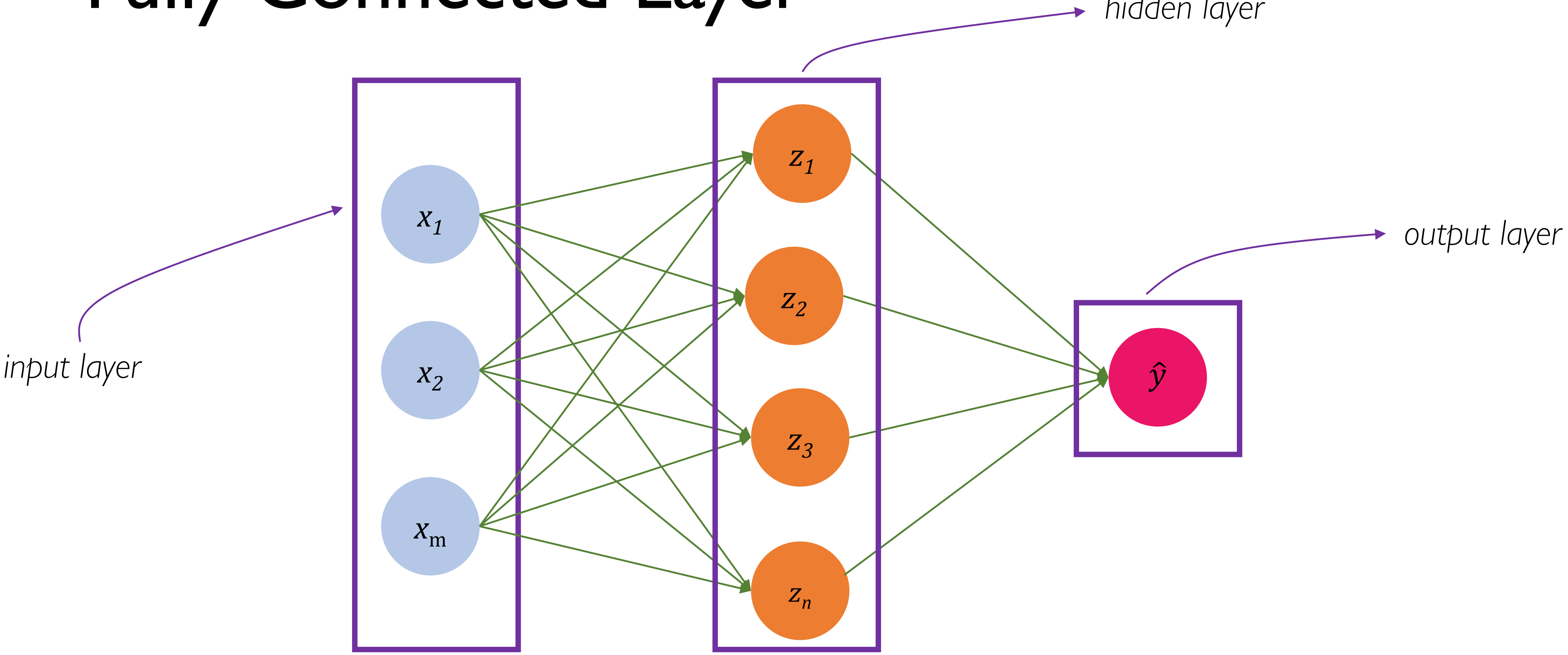
Jenis Neural Networks


A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

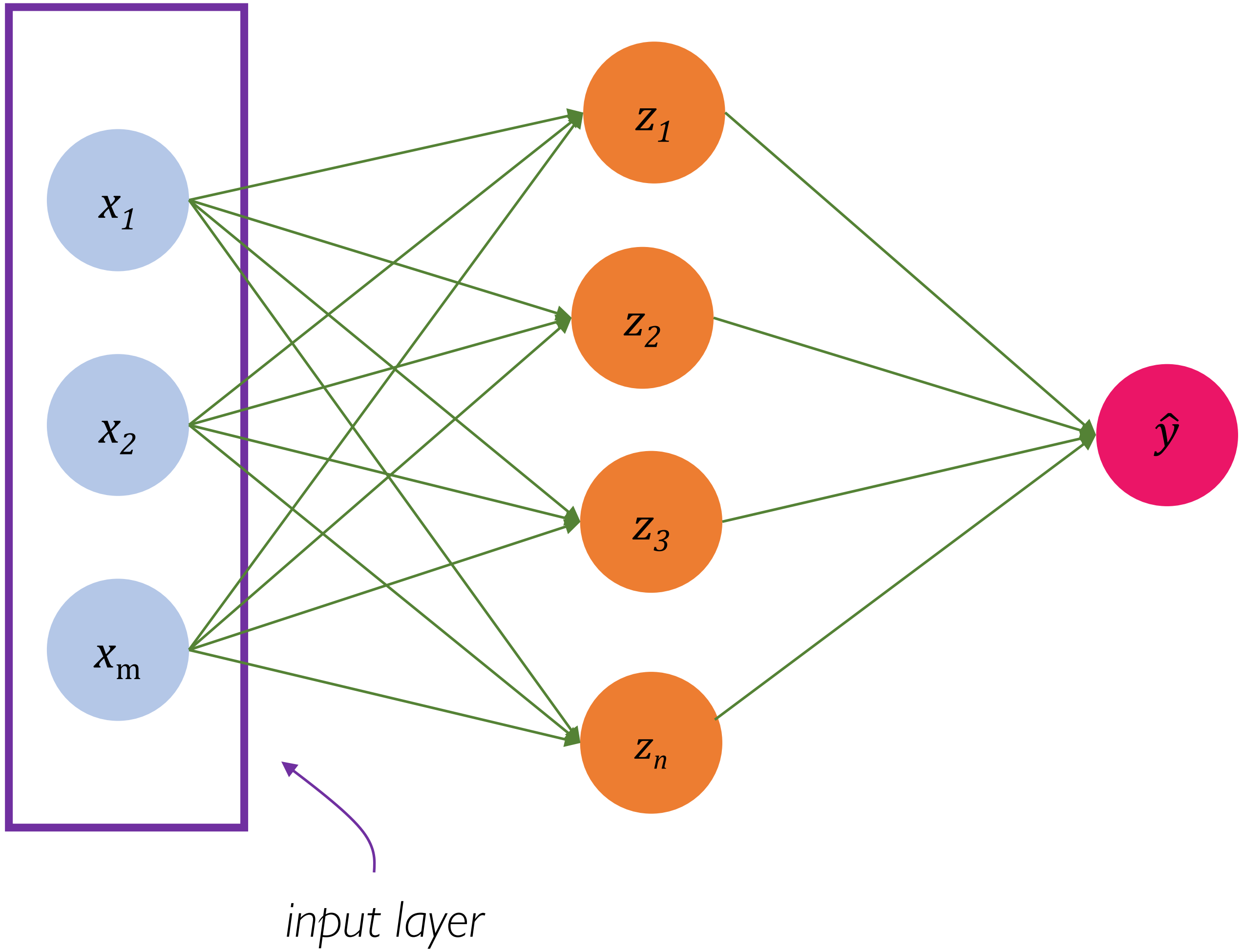
Fully Connected Layer



Fully Connected Layer



Fully Connected Layer

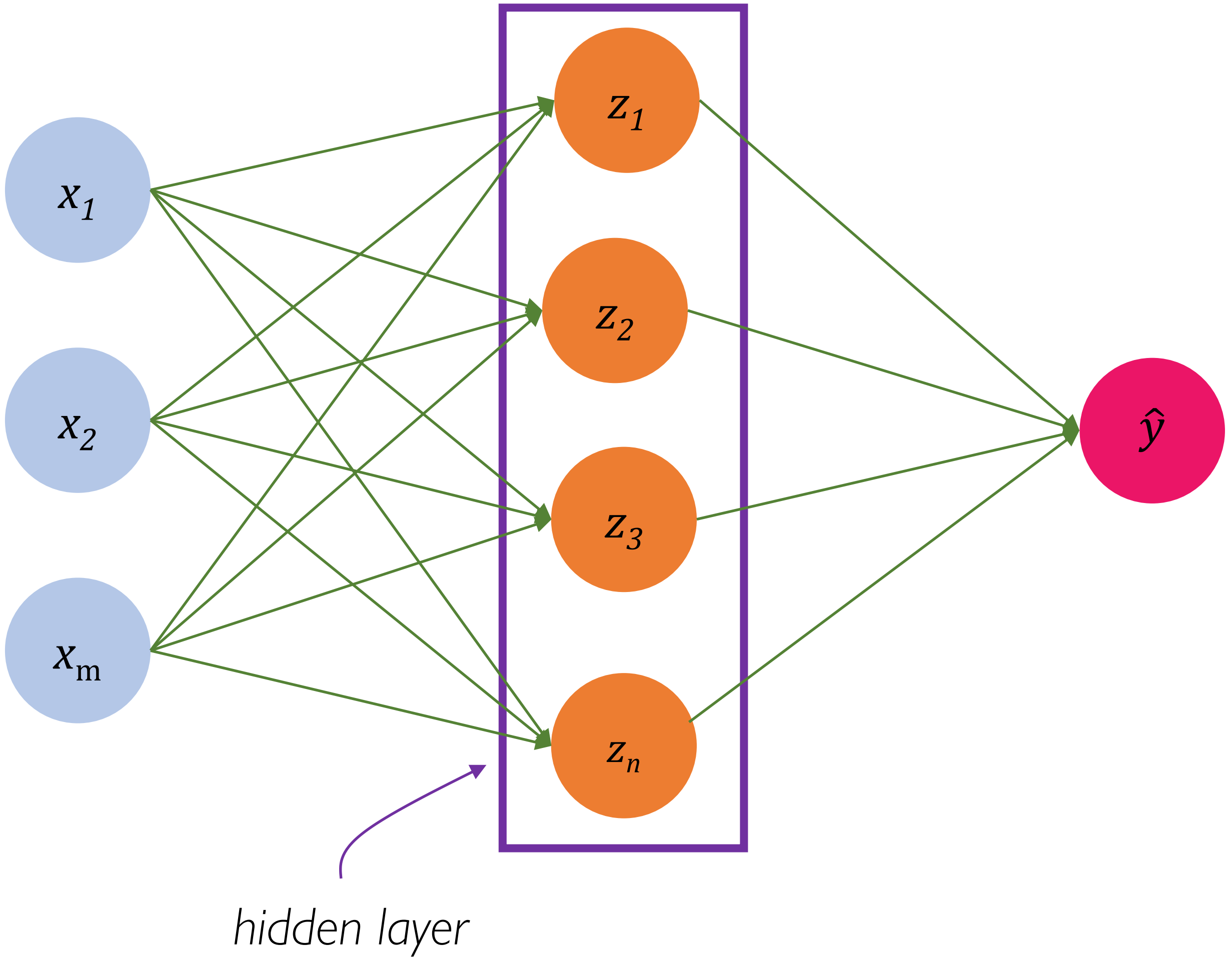


Input Layer

- Layer input menerima sinyal/nilai
- Jumlah neuron pada layer input sejumlah atribut pada data
- Atribut pada neuron input bertipe numerik ratio
- Nilai pada dataset harus dinormalisasi terlebih dahulu sebelum masuk ke jaringan
- Normalisasi menggunakan z-score, atau dengan kata lain nilai instance dikurangi dengan reratanya dan dibagi dengan standar deviasi

$$x' = \frac{x - \mu}{\sigma}$$

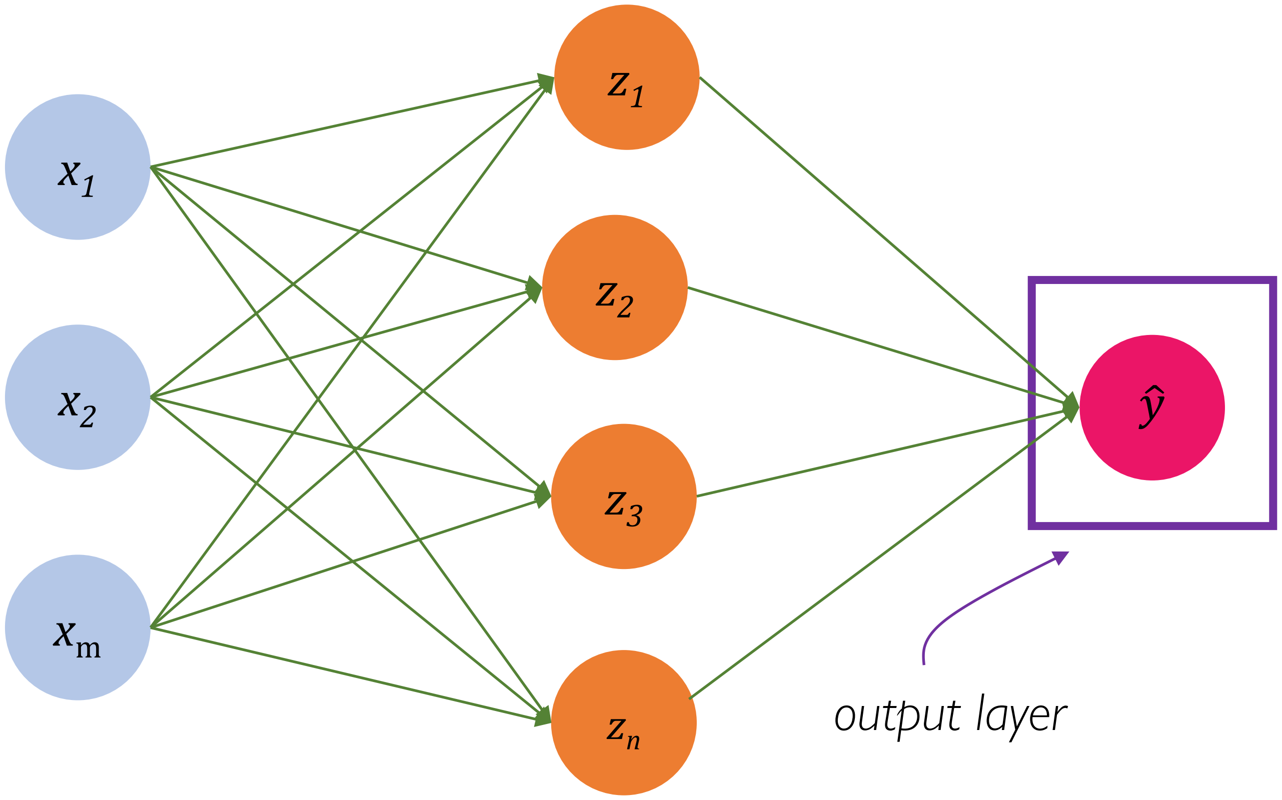
Fully Connected Layer



Hidden Layer

- Hidden layer terletak di antara layer input dan ouput
- Jumlah hidden layer dapat lebih dari 1 dengan jumlah node yang bebas
- Umumnya pada dataset yang tidak terlalu kompleks 1 hidden layer sudah cukup
- Hidden layer berfungsi menemukan pola non-linear pada data melalui activation function
- Activation function terdiri dari ReLU, Sigmoid, TanH dsb

Fully Connected Layer

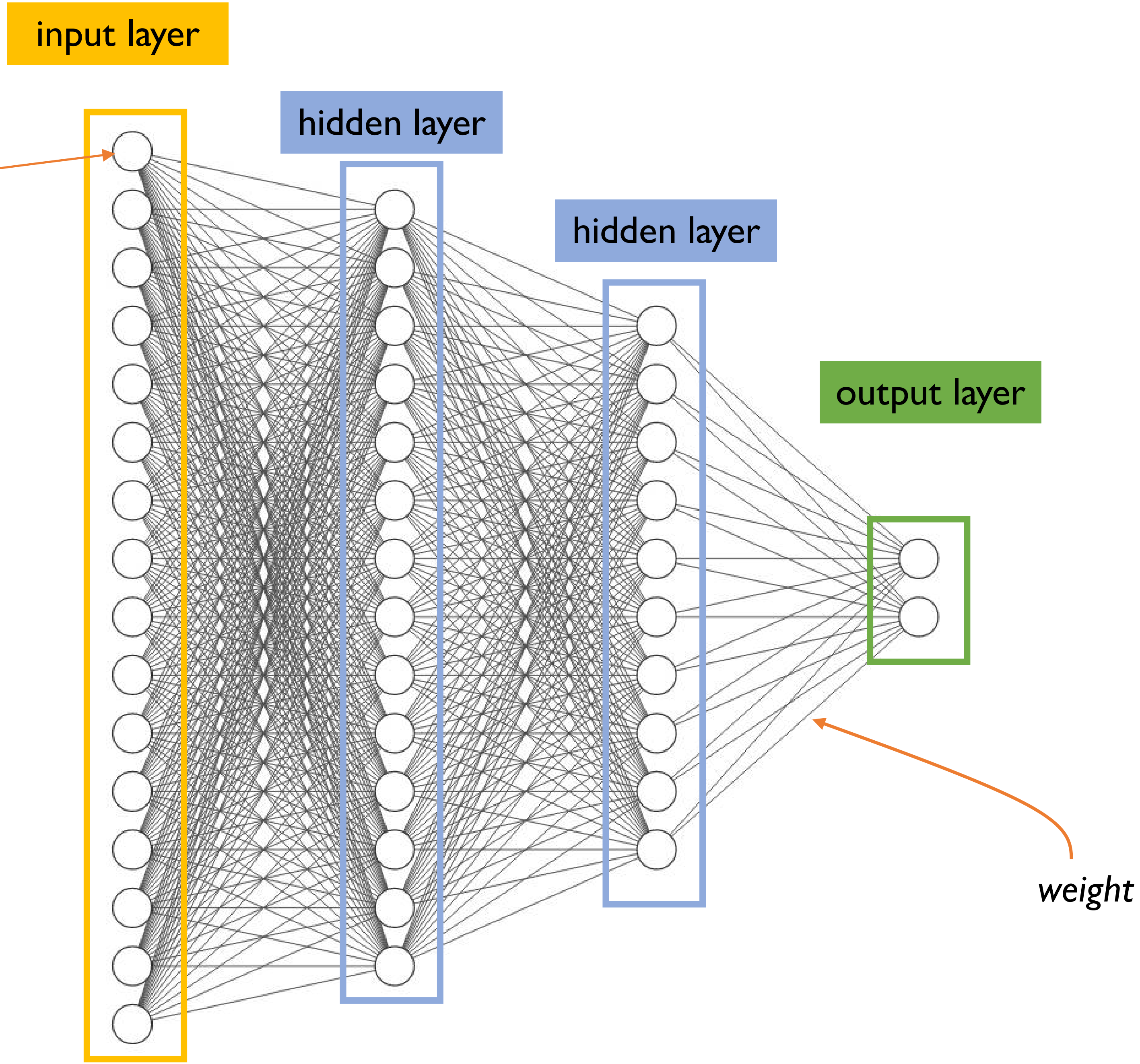


output layer

Output Layer

- Output layer mengeluarkan hasil prediksi dalam bentuk nilai. Dapat berupa label atau nilai encoding dari label
- Activation function pada output layer berbeda pada hidden layer, activation function pada layer ini berfungsi untuk menghasilkan nilai sesuai dengan encoding pada label
- Jenis activation function berupa sigmoid atau softmax

Fully Connected Layer



neuron $$y = \sum_{i=0}^{n}(x_i * \boldsymbol{w_i}) + \boldsymbol{b_i}$$

activation function $$f(x) = max(0, x)$$

loss function $$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

backpropagation $$\frac{\partial f}{\partial w_{jk}^i} = \frac{\partial f}{\partial z_j^i}\frac{\partial z_j^i}{\partial w_{jk}^i}$$

CONTOH

Contoh Permasalahan

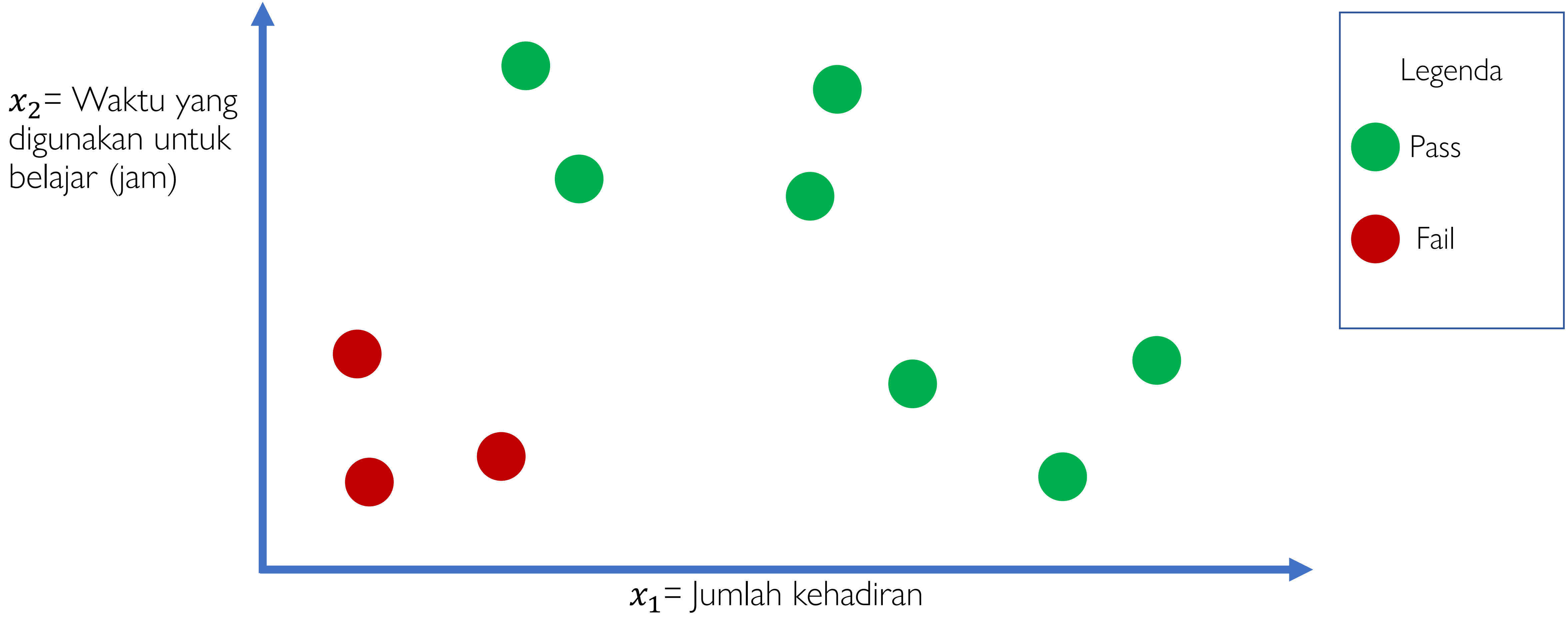
Apakah saya akan lulus dari mata kuliah ini?

Kita mulai dengan dua fitur yang sederhana

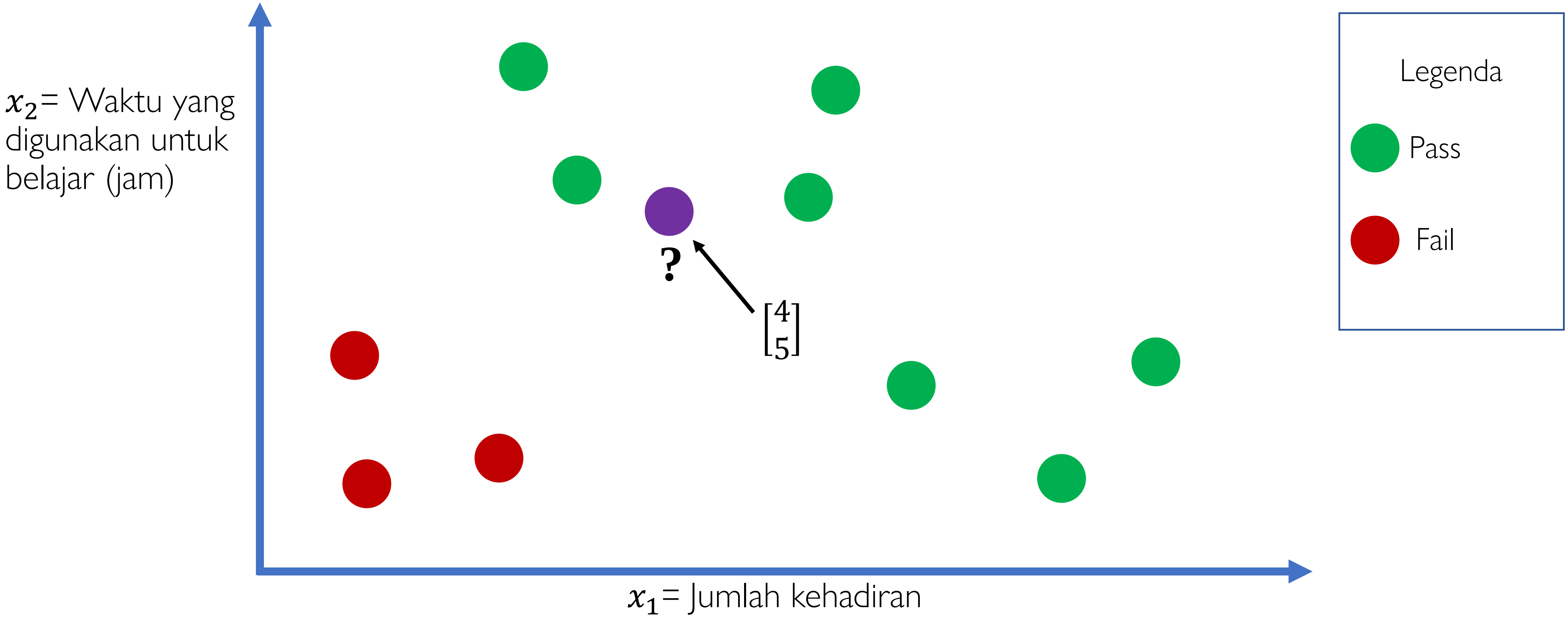
x_1= Jumlah kehadiran

x_2= Waktu yang digunakan untuk belajar (jam)

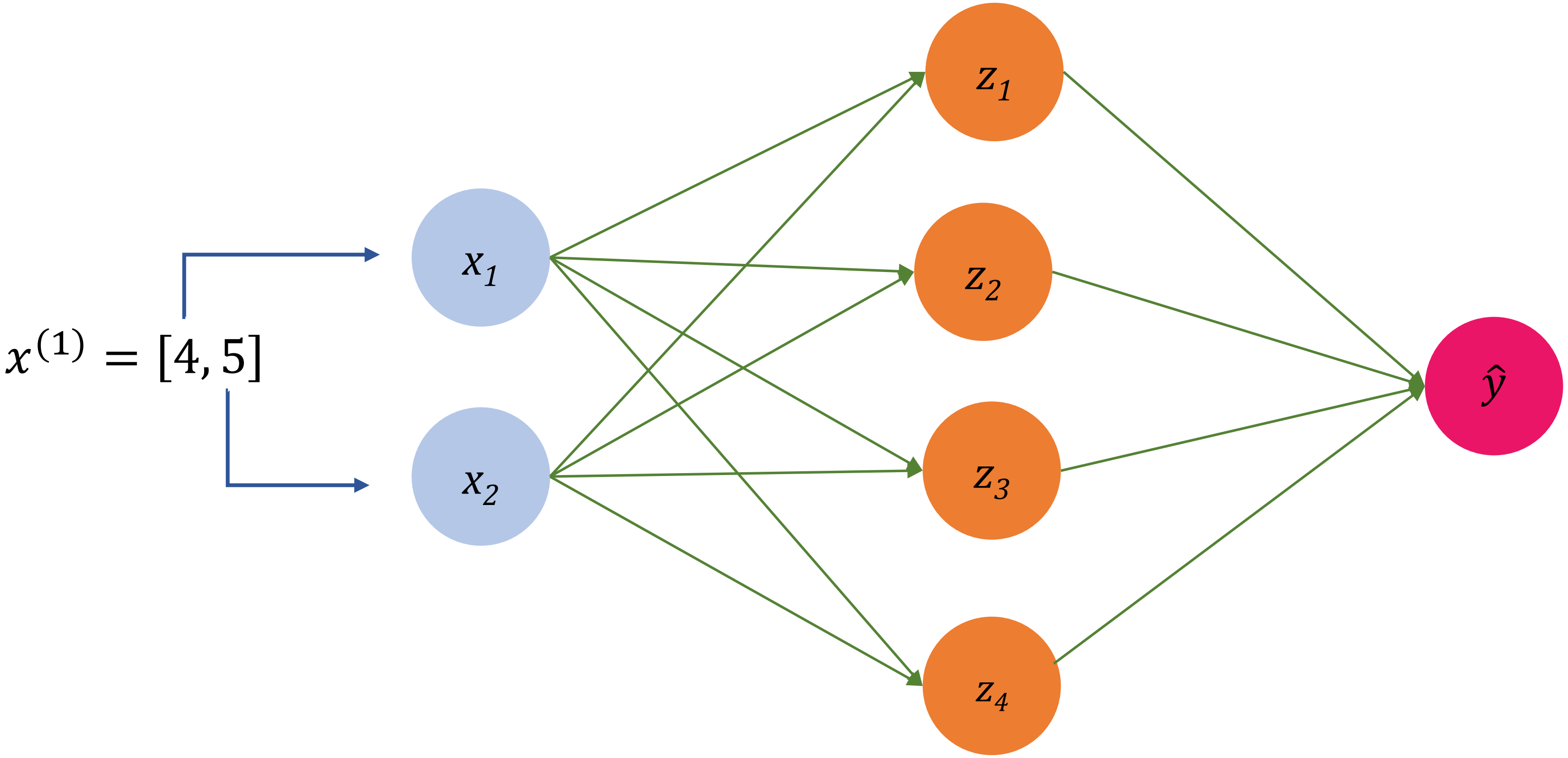
Contoh Permasalahan



Contoh Permasalahan

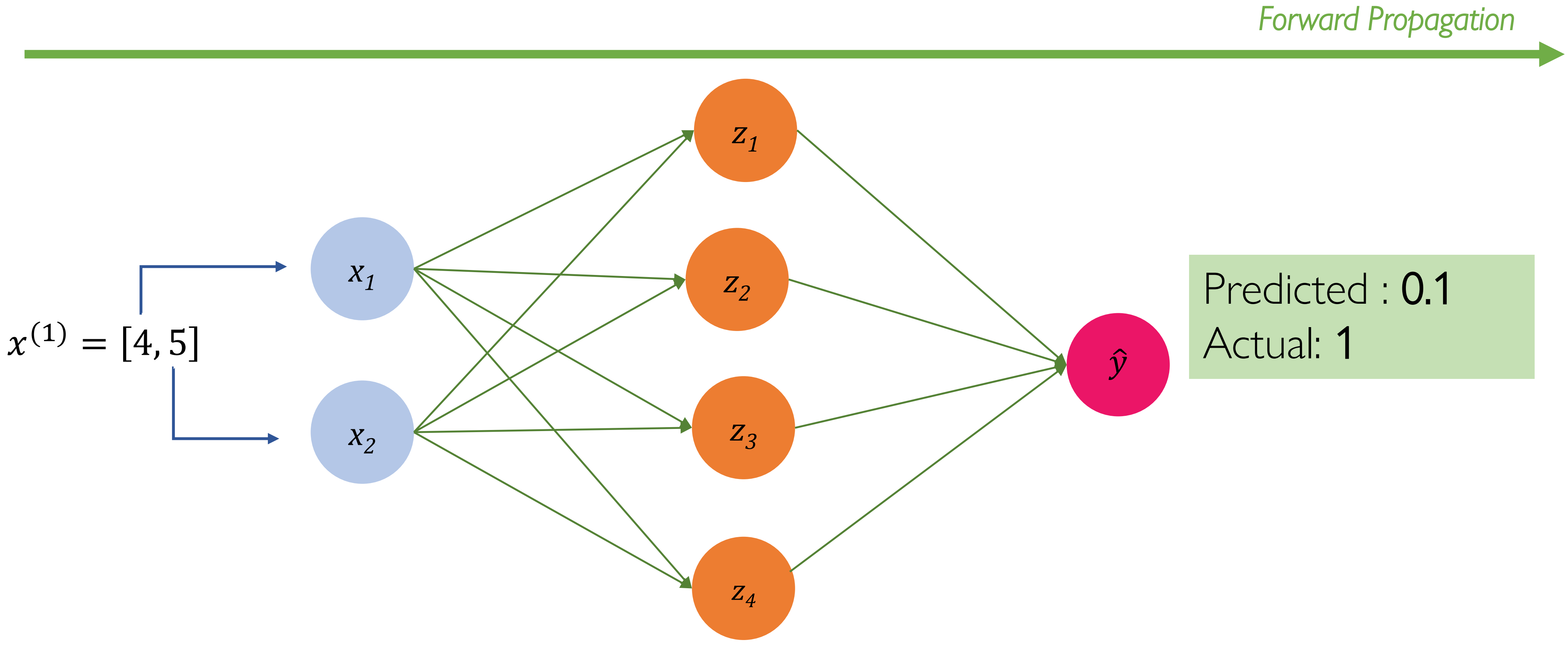


Contoh Permasalahan



Predicted : 0.1
Actual: 1

Forward Propagation



$x^{(1)} = [4, 5]$

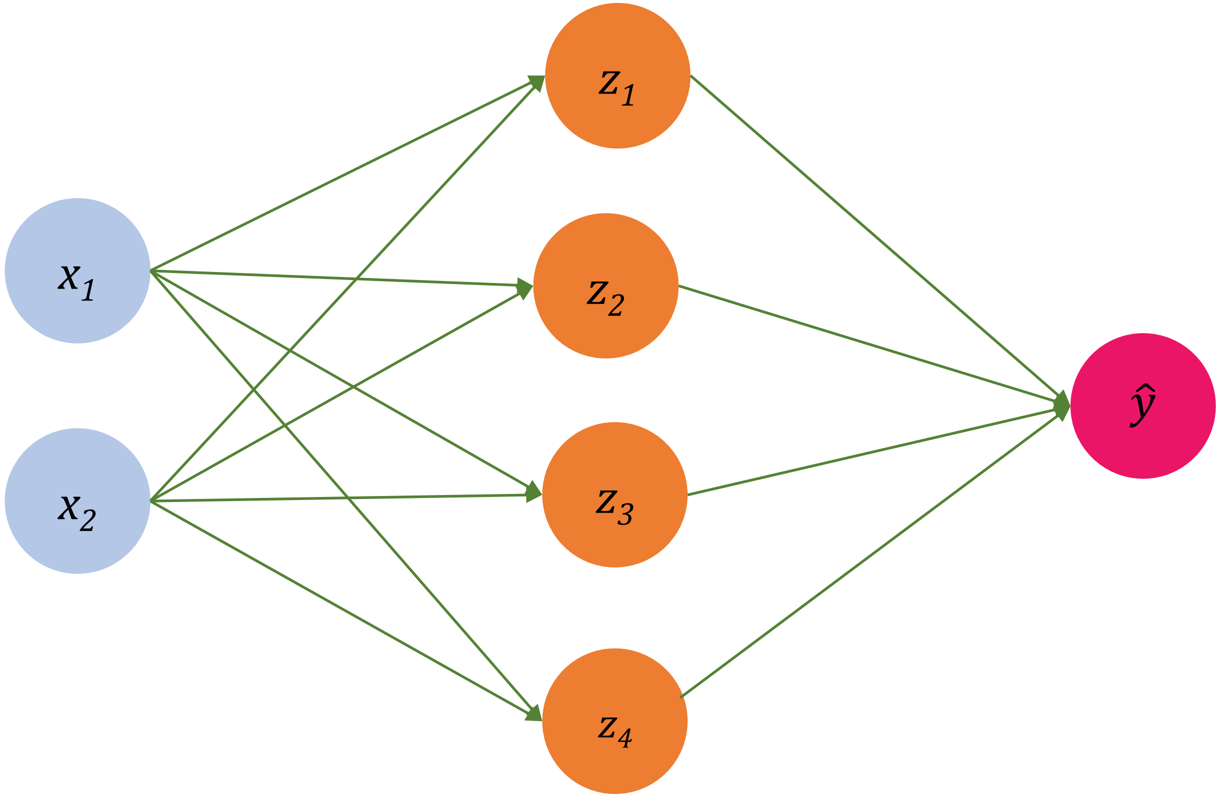
Predicted : 0.1
Actual: 1

$$\mathcal{L}\left(f\left(x^{(i)}; W\right), y^{(i)}\right)$$

Batch Processing

Forward Propagation

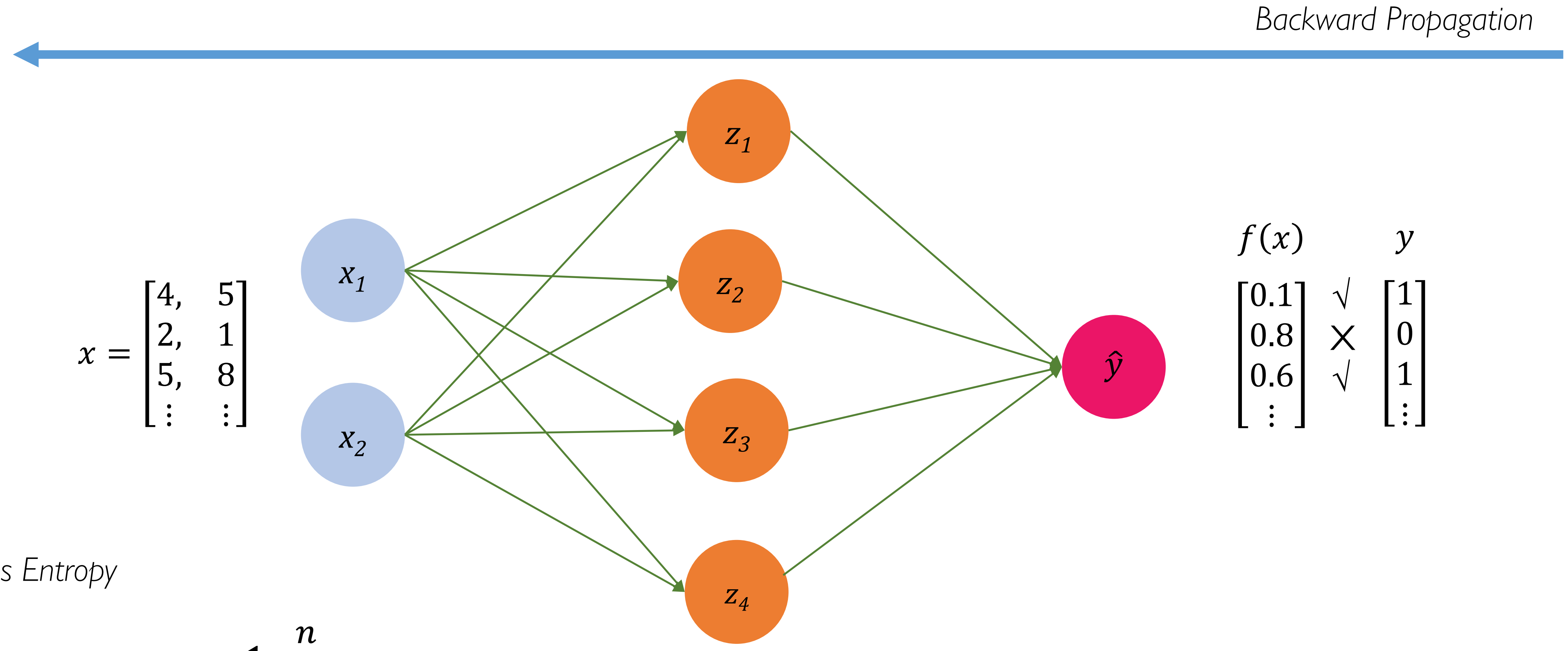
$$x = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$$\begin{array}{ccc} f(x) & & y \\ \begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} & \begin{matrix} \surd \\ \times \\ \surd \\ \end{matrix} & \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \end{array}$$

$$\mathcal{L}\Big(\underbrace{f\big(x^{(i)}; W\big)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\Big)$$

Loss Function



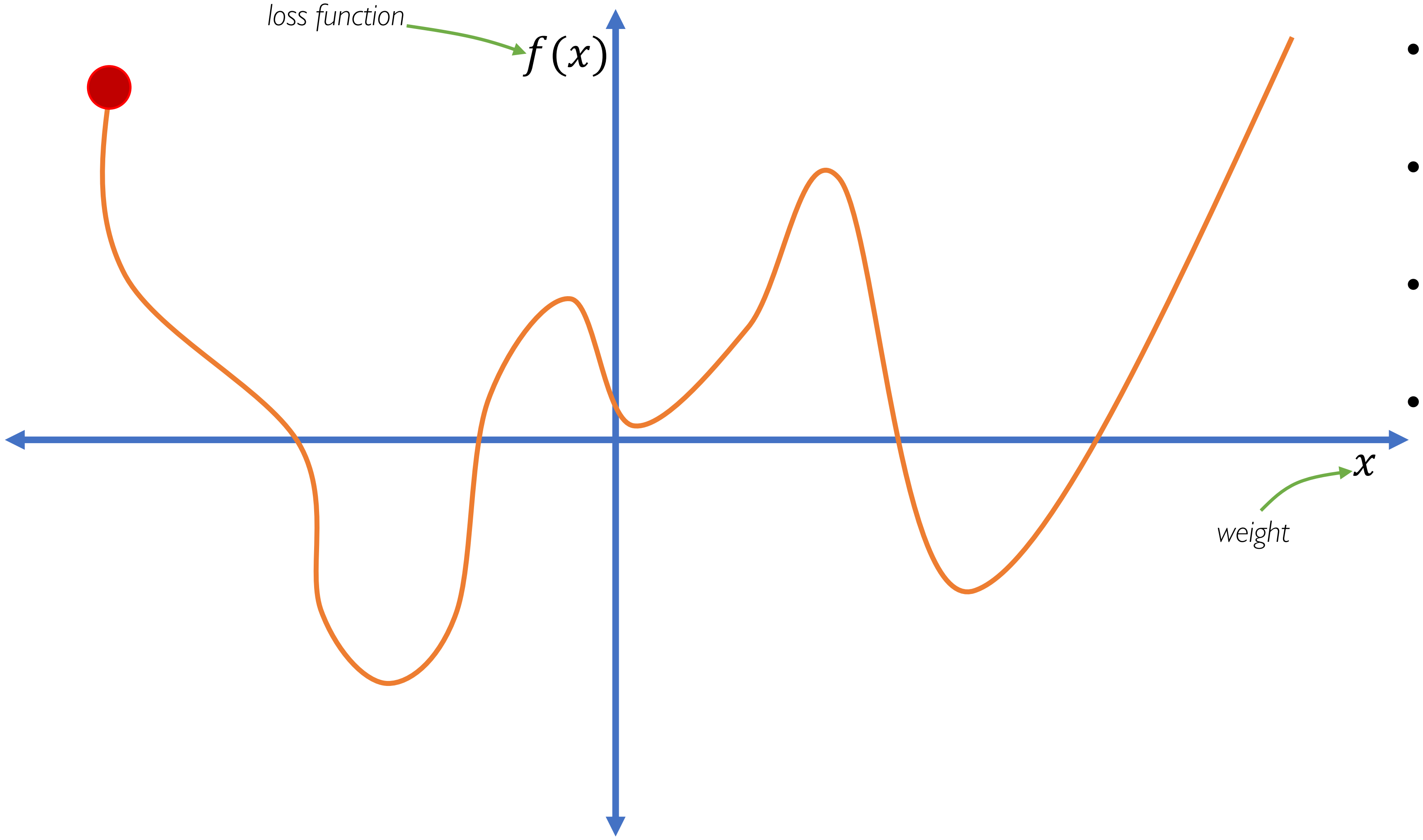
$$x = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$

$$\begin{array}{ccc} f(x) & & y \\ \begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} & \begin{array}{c} \surd \\ \times \\ \surd \\ \\ \end{array} & \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \end{array}$$

Binary Cross Entropy

$$\boldsymbol{J}(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\left(\underbrace{f\left(x^{(i)}; W\right)}_{\text{Predicted}}\right) + \left(1 - \underbrace{y^{(i)}}_{\text{Actual}}\right) \log\left(1 - \underbrace{f\left(x^{(i)}; W\right)}_{\text{Predicted}}\right)$$

Loss Minimization

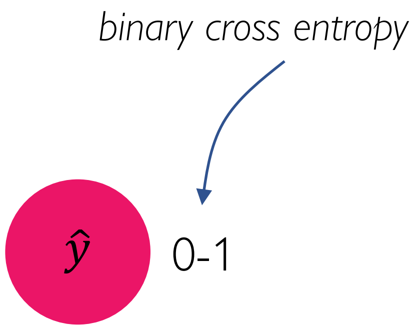


Berbagai jenis algoritma learning:

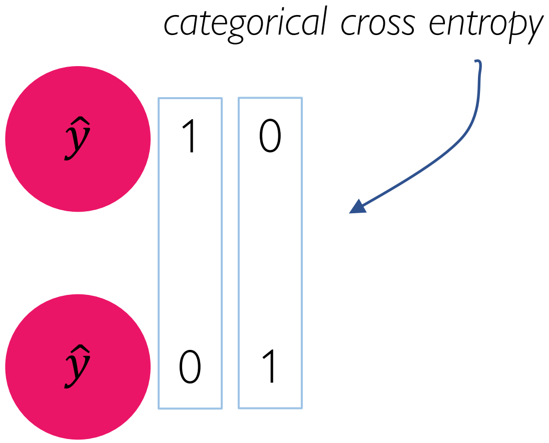
- SGD: Stochastic Gradient Descent
- Adam: Adaptive Moment Estimation
- RMSProp: Root Mean Square Propagation
- dst…

Loss Function & Label Encoding

Binary class (2 kelas label)

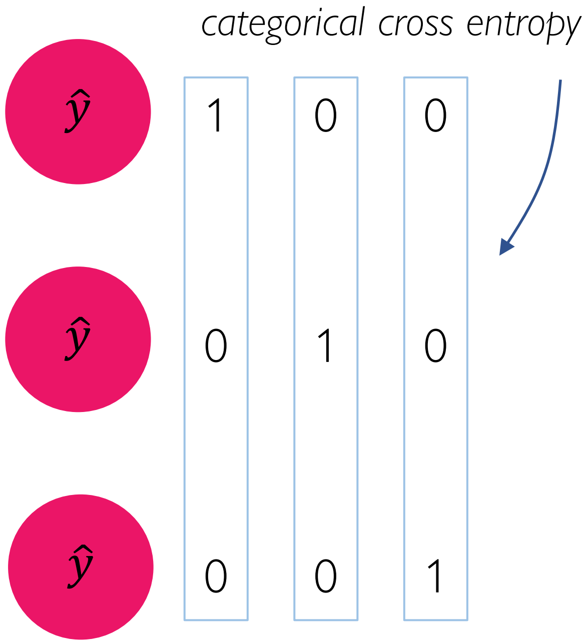


Kelas 1 mendekati 0
Kelas 2 mendekati 1



Kelas 1 [1 0]
Kelas 2 [0 1]

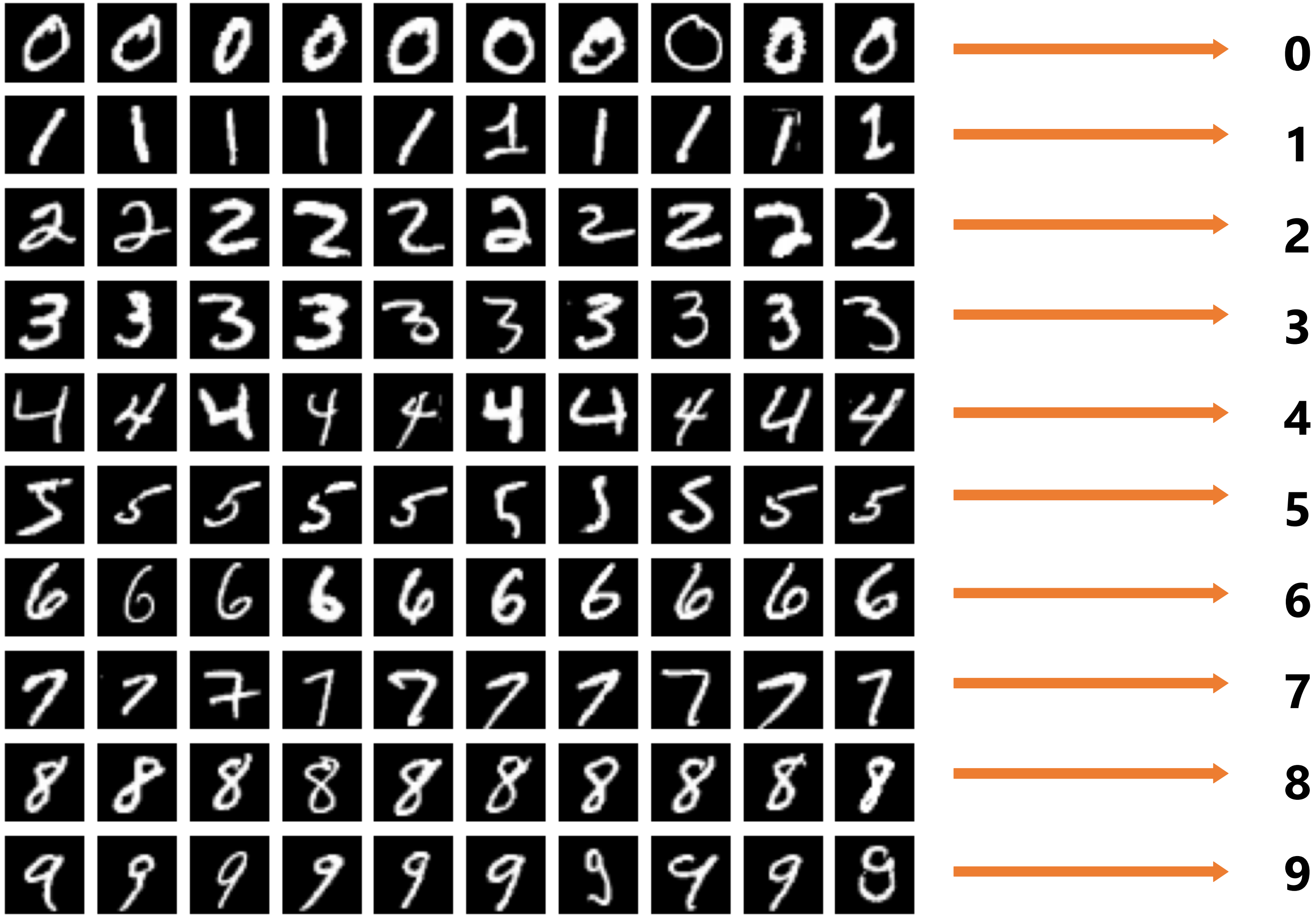
Multi label (lebih dari 2 kelas label)



Kelas 1 [1 0 0]
Kelas 2 [0 1 0]
Kelas 3 [0 0 1]

IMPLEMENTASI

Handwriting Recognition



0

1

2

3

4

5

6

7

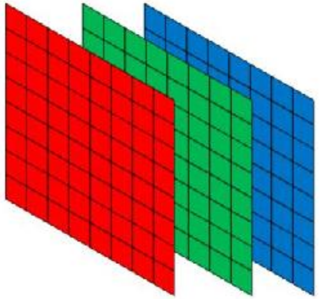
8

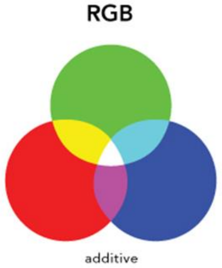
9

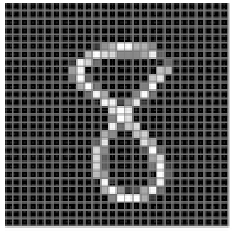
bagaimana caranya mengajari komputer untuk mengenal angka?

Citra Digital

12	99	167	240	255	242	58	58
53	100	85	86	82	75	35	75
58	101	230	240	86	27	29	89
65	247	244	78	98	59	200	80
68	185	87	158	78	178	201	97
86	175	178	89	78	78	235	111
244	168	88	175	48	78	225	121
230	124	168	125	135	148	240	121





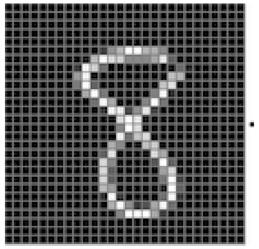


28 x 28
784 pixels

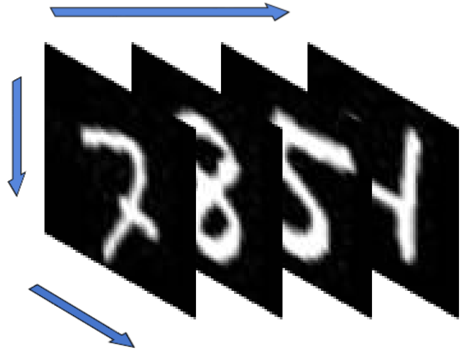
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	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

citra digital memiliki tipe data uint8, sehingga nilainya adalah 0-255

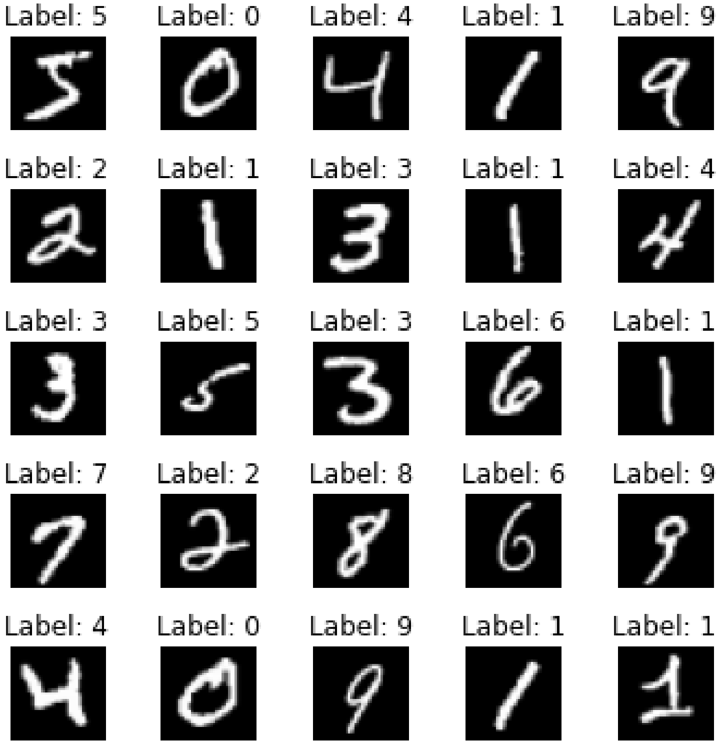
MNIST Dataset



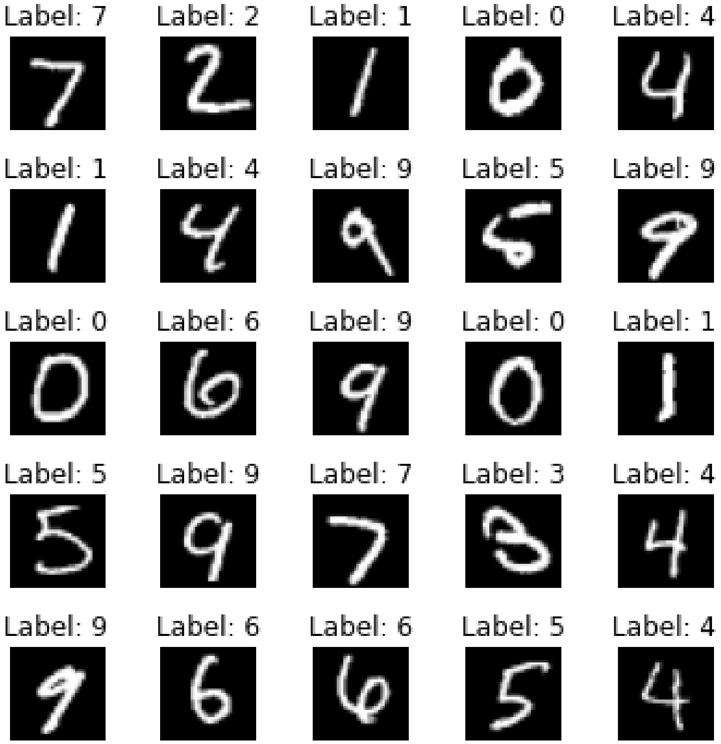
28 x 28
784 pixels



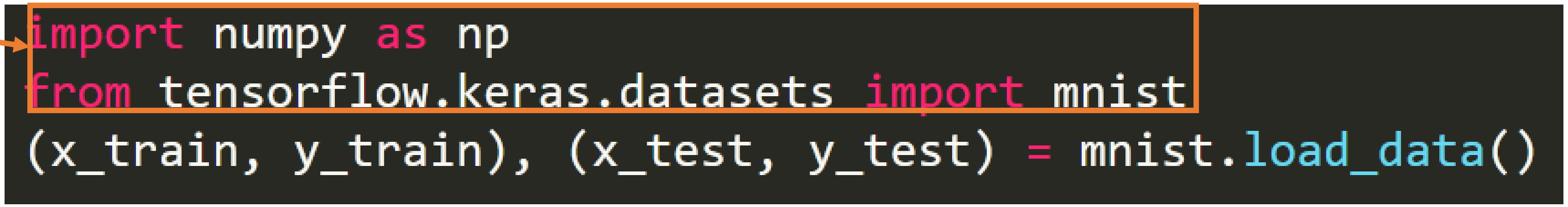
Training 60.000



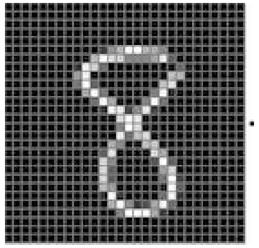
Testing 10.000



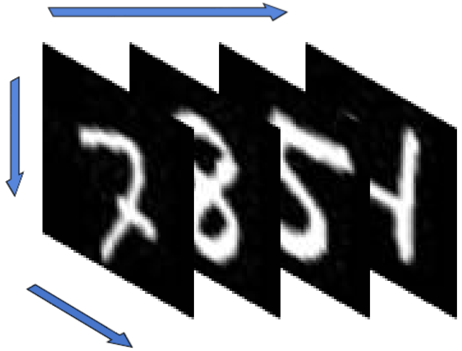
load library



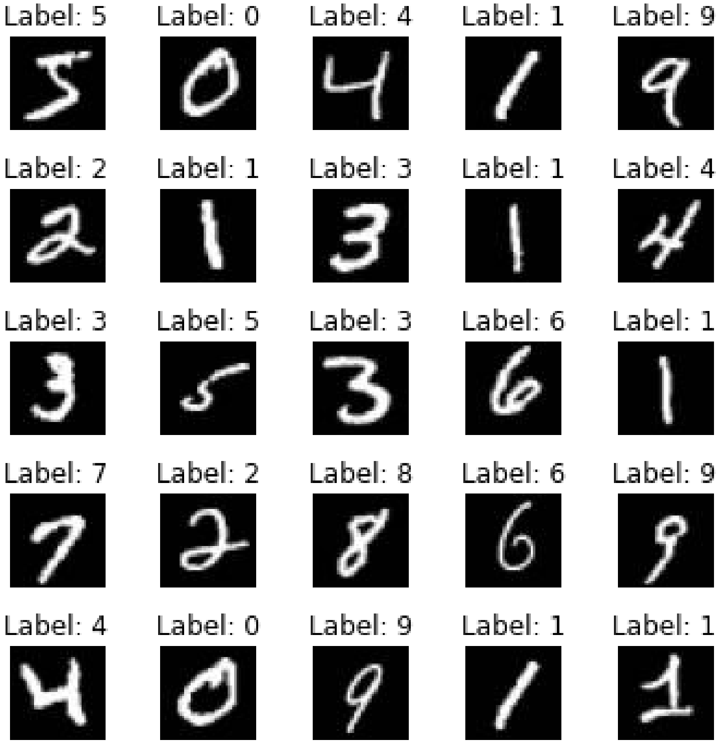
```
import numpy as np
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```



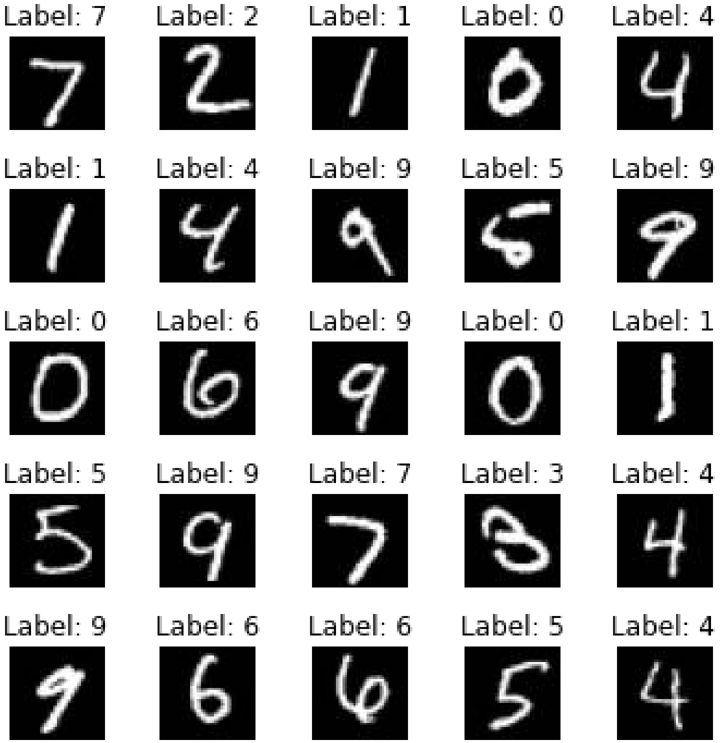
28 x 28
784 pixels



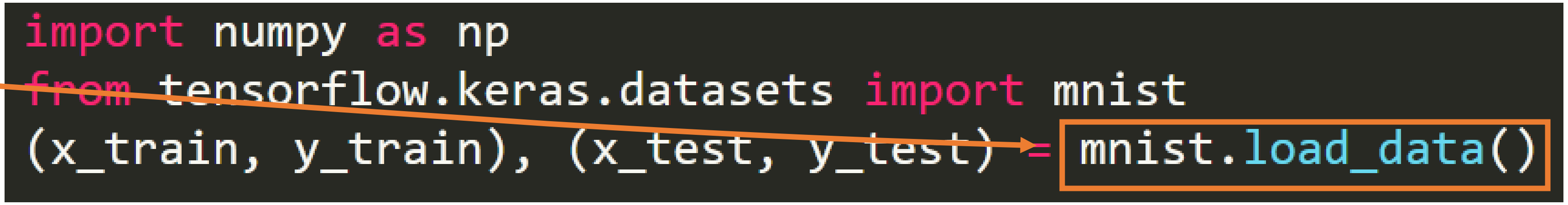
Training 60.000



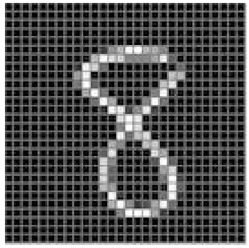
Testing 10.000



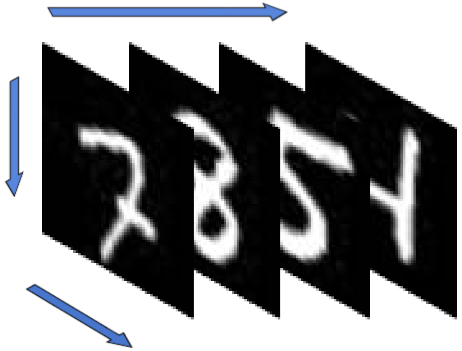
load mnist data



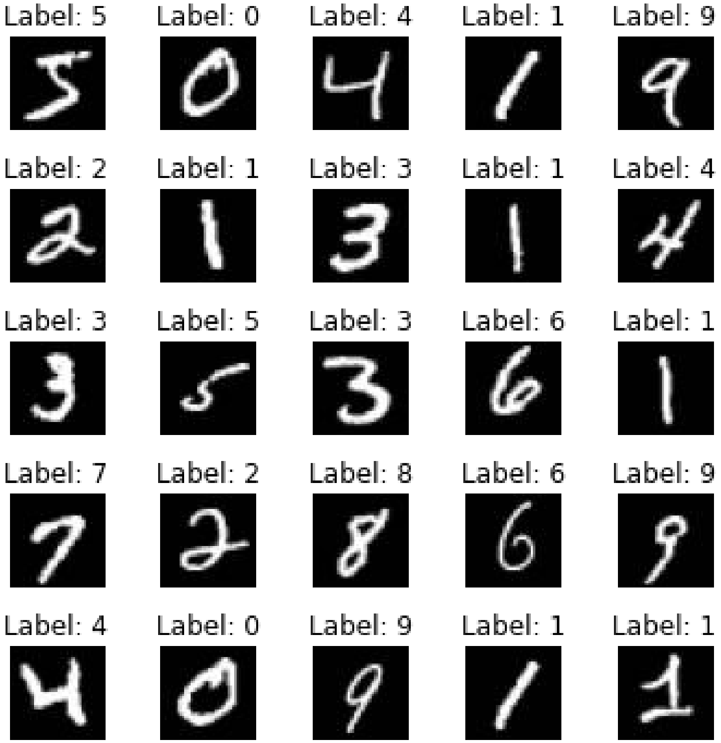
```
import numpy as np
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```



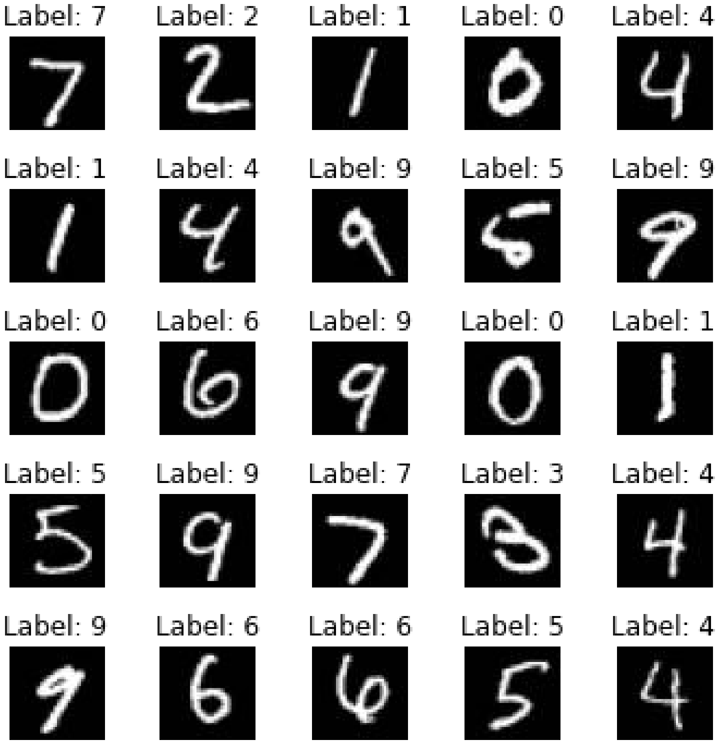
28 x 28
784 pixels



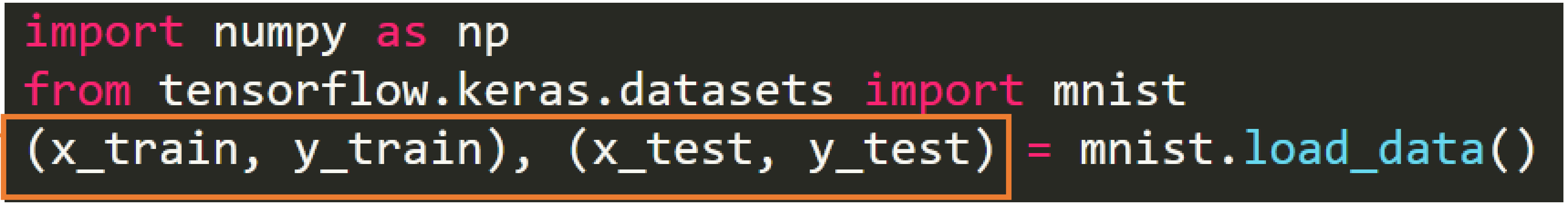
Training 60.000



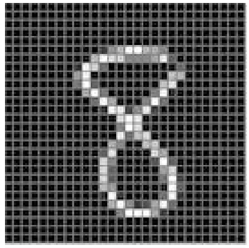
Testing 10.000



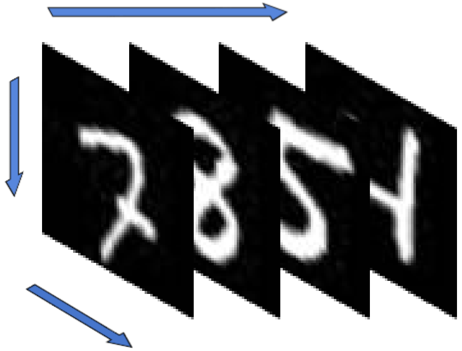
unpack value



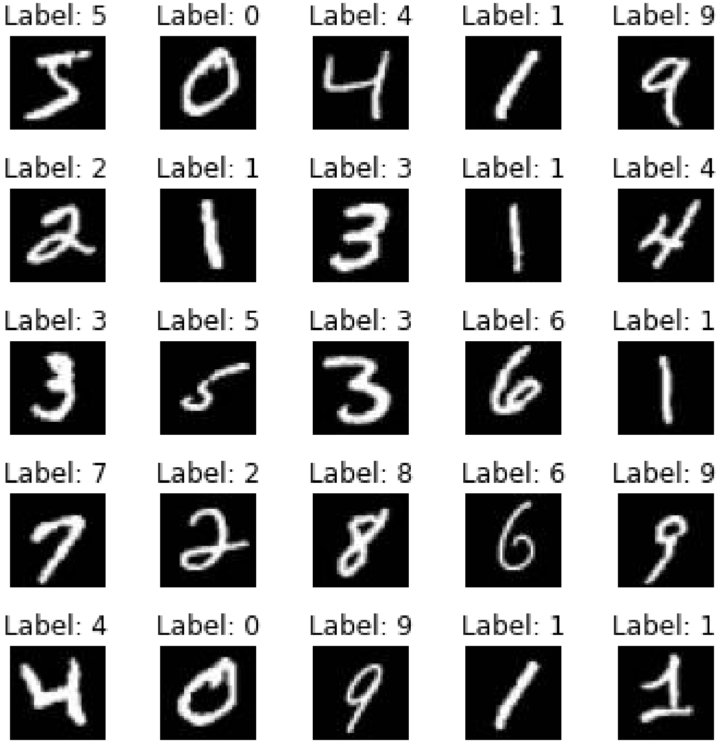
```
import numpy as np
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```



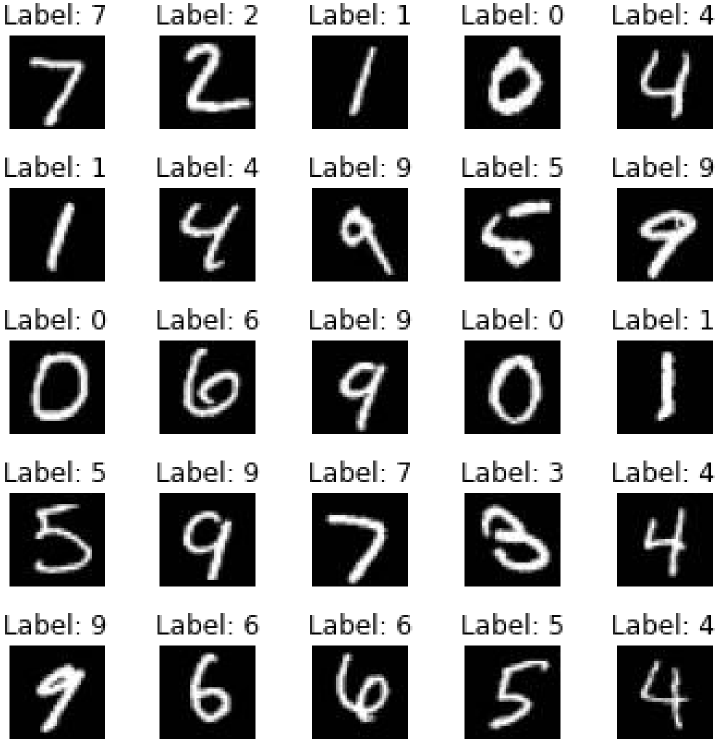
28 x 28
784 pixels

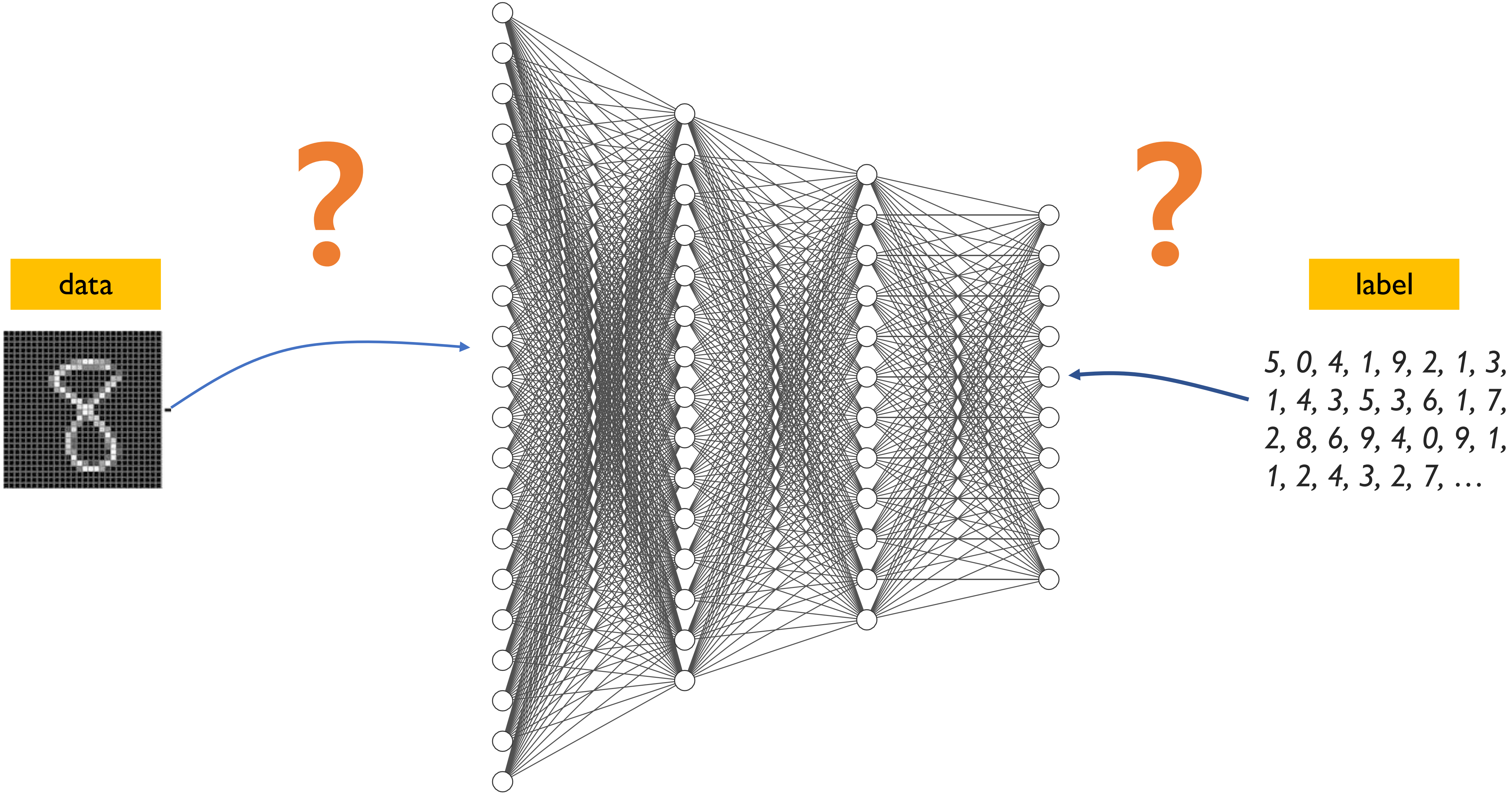


Training 60.000



Testing 10.000

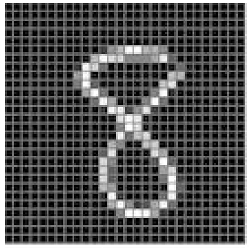




data

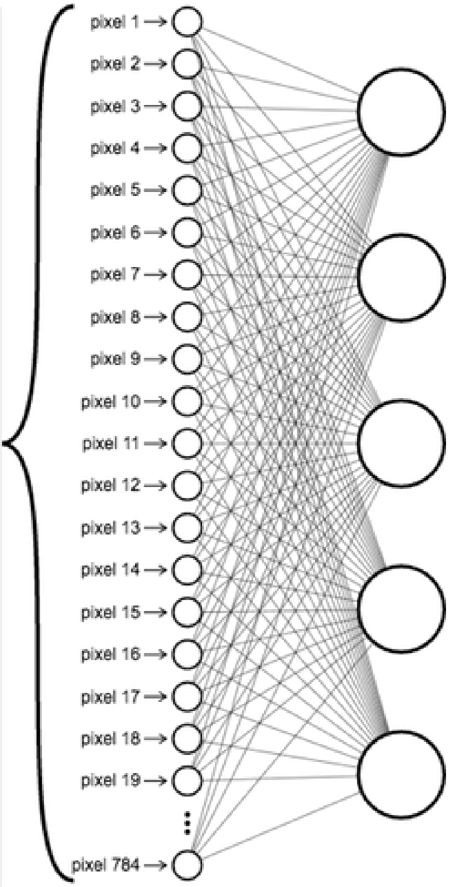
label

5, 0, 4, 1, 9, 2, 1, 3,
1, 4, 3, 5, 3, 6, 1, 7,
2, 8, 6, 9, 4, 0, 9, 1,
1, 2, 4, 3, 2, 7, …



28 x 28
784 pixels

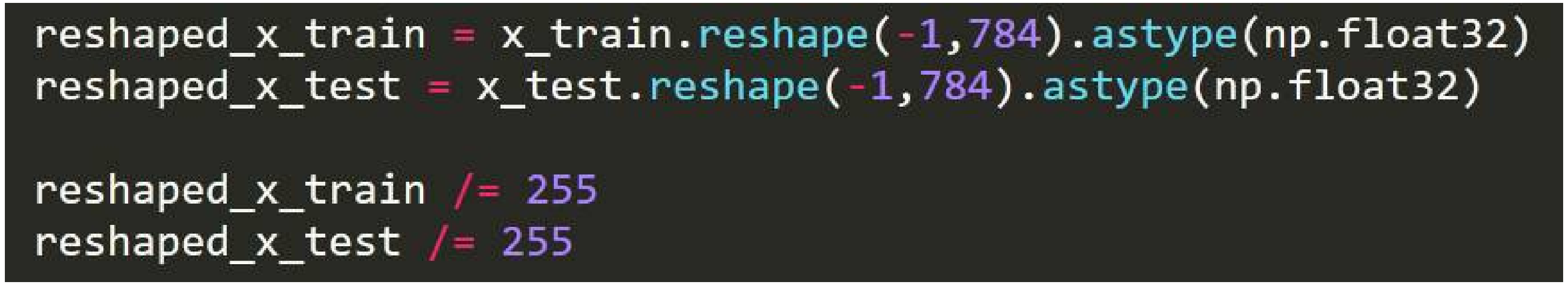
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	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



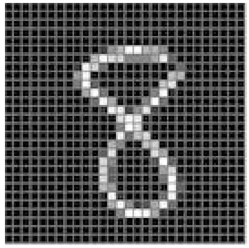
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\Downarrow$$

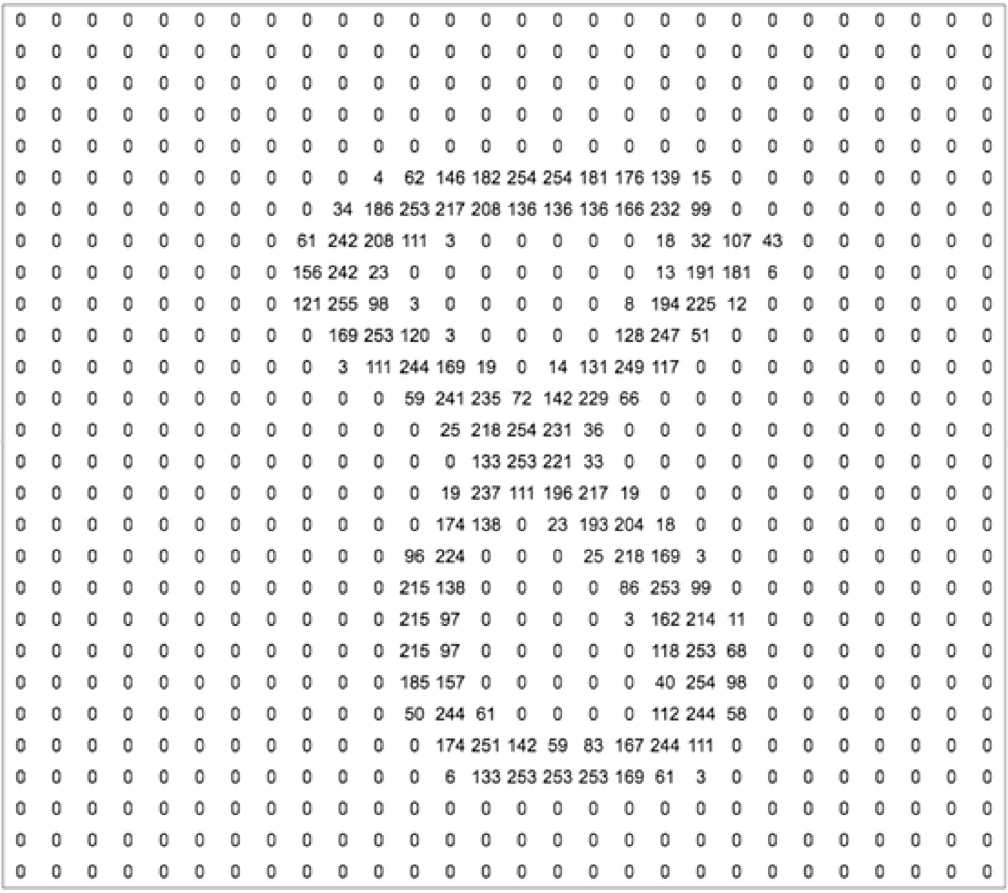
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$



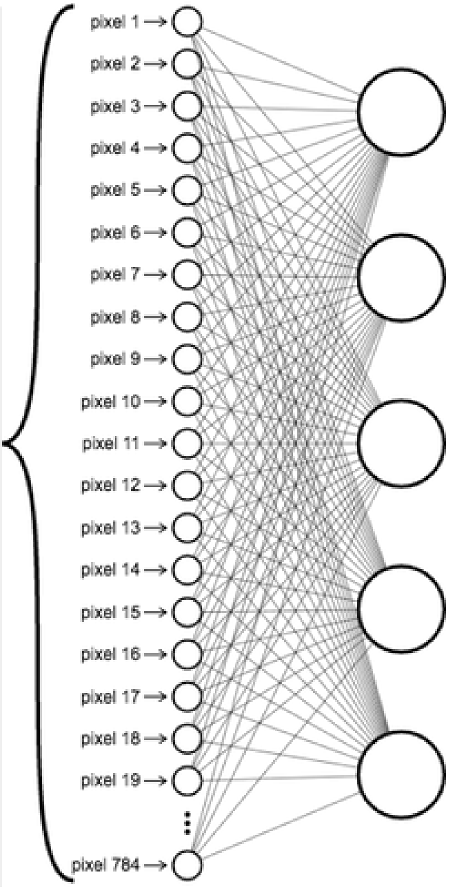
```
reshaped_x_train = x_train.reshape(-1,784).astype(np.float32)
reshaped_x_test = x_test.reshape(-1,784).astype(np.float32)

reshaped_x_train /= 255
reshaped_x_test /= 255
```



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

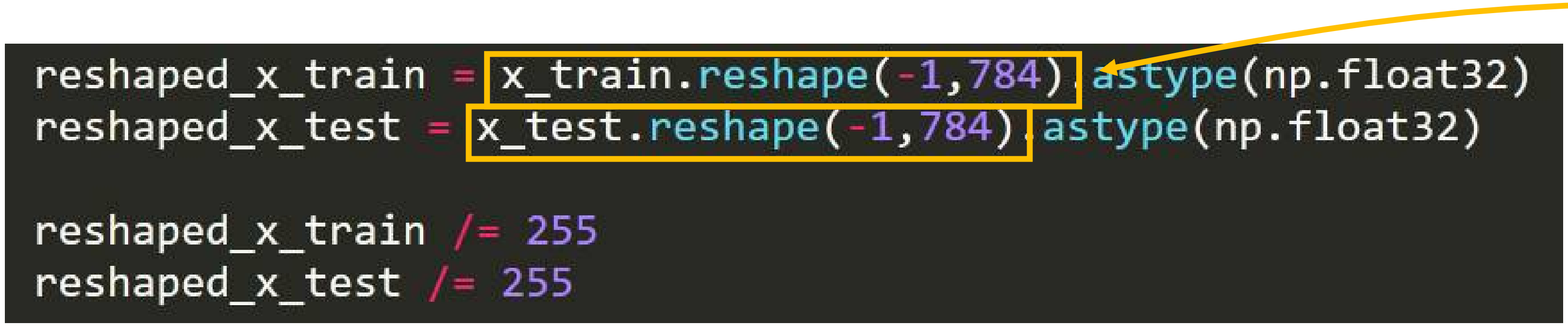


$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

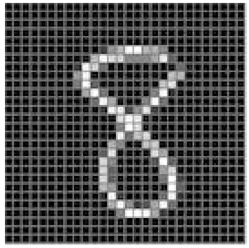
$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

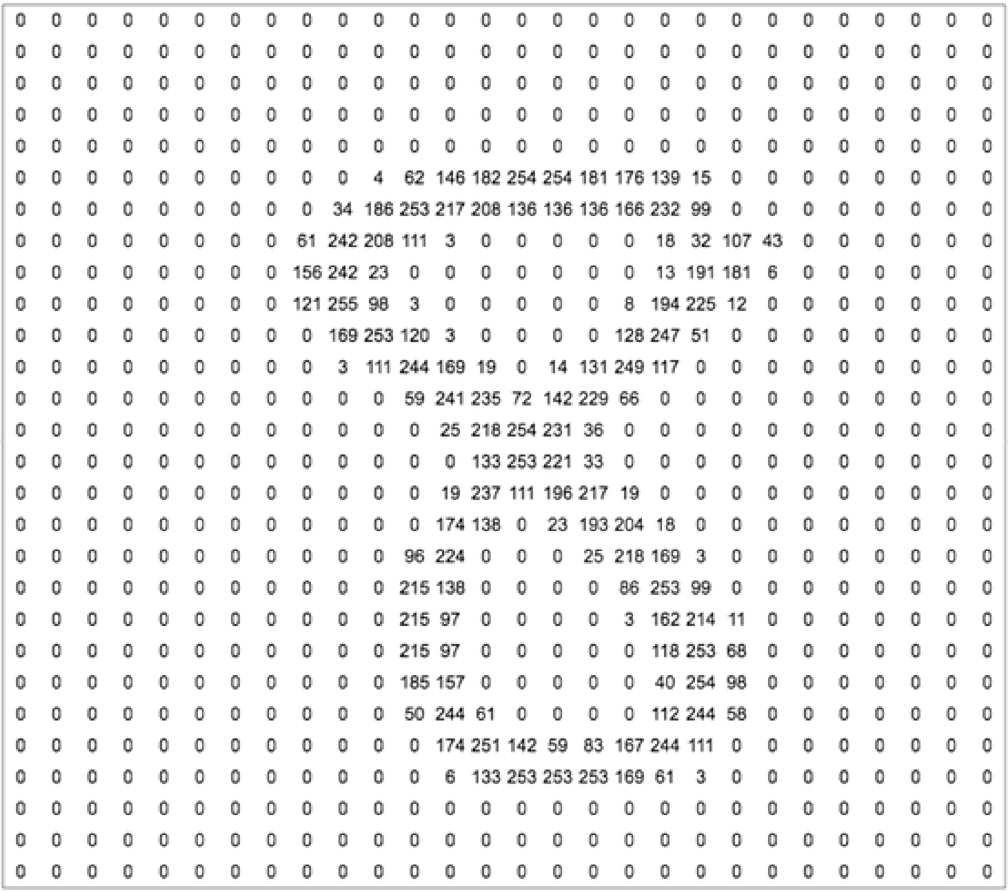
ubah shape/bentuk data



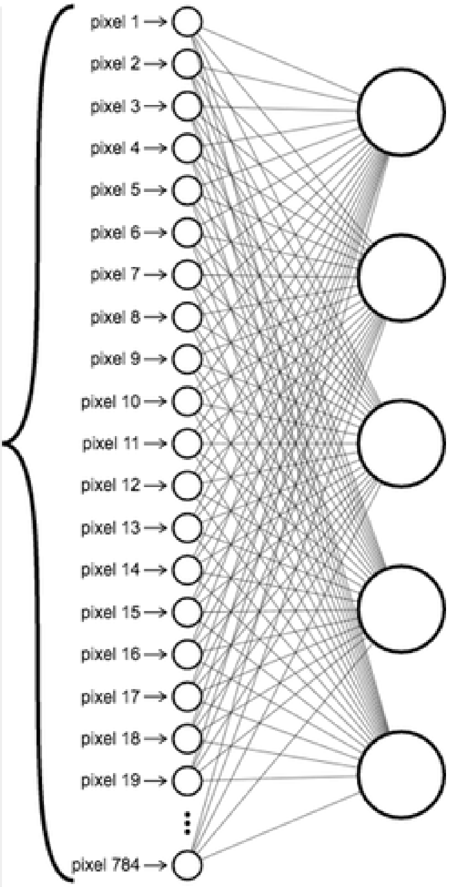
```
reshaped_x_train = x_train.reshape(-1,784).astype(np.float32)
reshaped_x_test = x_test.reshape(-1,784).astype(np.float32)

reshaped_x_train /= 255
reshaped_x_test /= 255
```



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

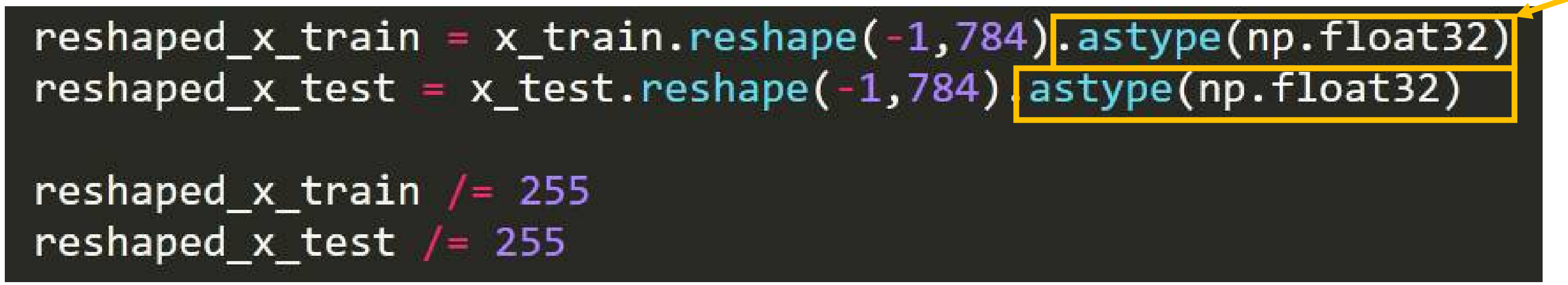


$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

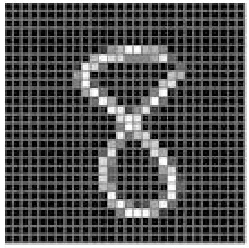
$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

ubah tipe data



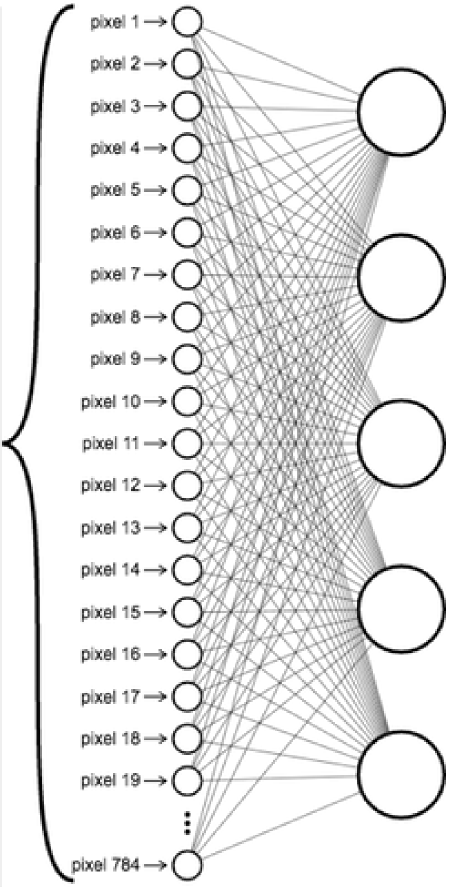
```
reshaped_x_train = x_train.reshape(-1,784).astype(np.float32)
reshaped_x_test = x_test.reshape(-1,784).astype(np.float32)

reshaped_x_train /= 255
reshaped_x_test /= 255
```



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

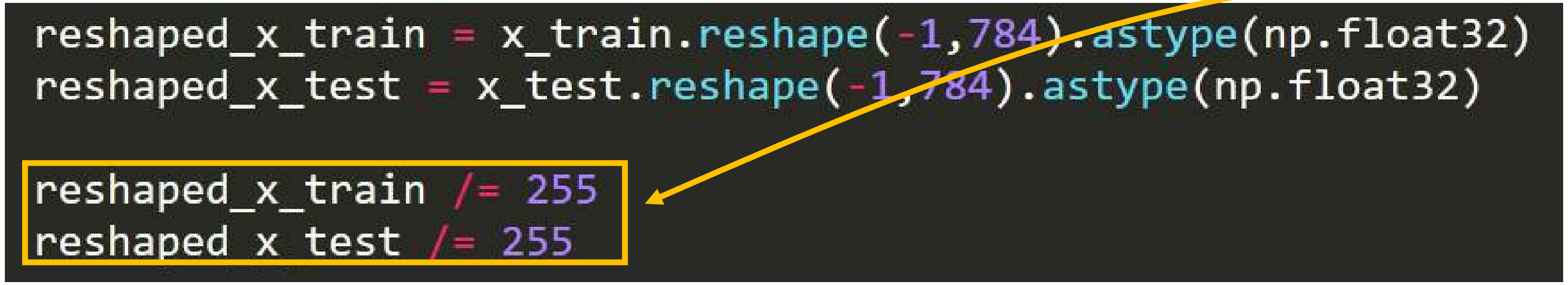


$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\Big\Downarrow$$

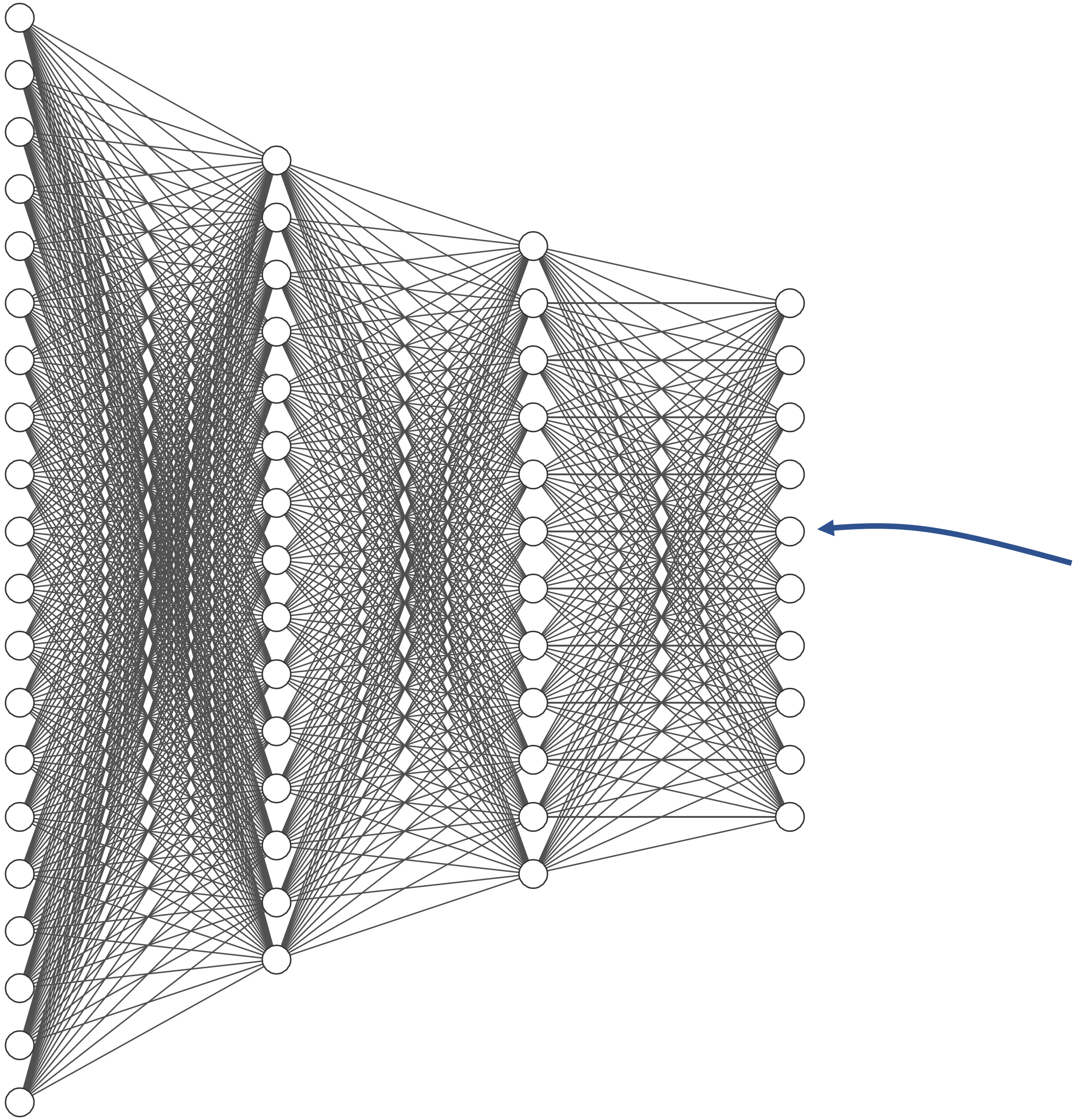
$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

normalisasi nilai



```
reshaped_x_train = x_train.reshape(-1,784).astype(np.float32)
reshaped_x_test = x_test.reshape(-1,784).astype(np.float32)

reshaped_x_train /= 255
reshaped_x_test /= 255
```

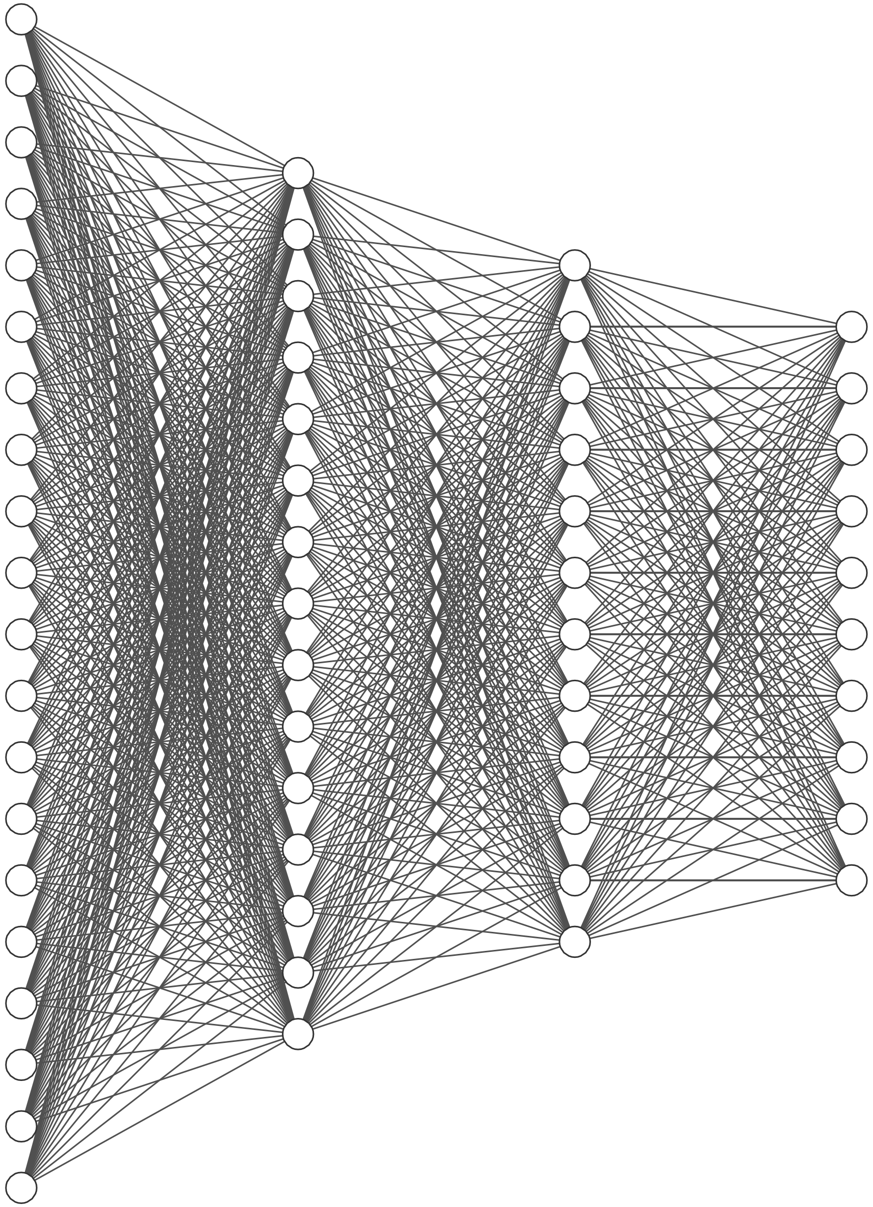
nilai piksel akan memiliki rentang 0-1
hal ini akan memudahkan model dalam training



label

5, 0, 4, 1, 9, 2, 1, 3,
1, 4, 3, 5, 3, 6, 1, 7,
2, 8, 6, 9, 4, 0, 9, 1,
1, 2, 4, 3, 2, 7, …

0 = [_ _   _ _ _ _ _ _ _]
1 = [_ _ _ _ _ _ _ _ _ _]
2 = [_ _ _ _ _ _ _ _ _]
3 = [_ _ _ _ _ _ _ _ _ _]
4 = [_ _ _ _ _ _ _ _ _ _]
5 = [_ _ _ _ _ _ _ _ _ _]
6 = [_ _ _ _ _ _ _ _ _ _]
7 = [_ _ _ _ _ _ _ _ _ _]
8 = [_ _ _ _ _ _ _ _ _ _]
9 = [_ _ _ _ _ _ _ _ _ _]

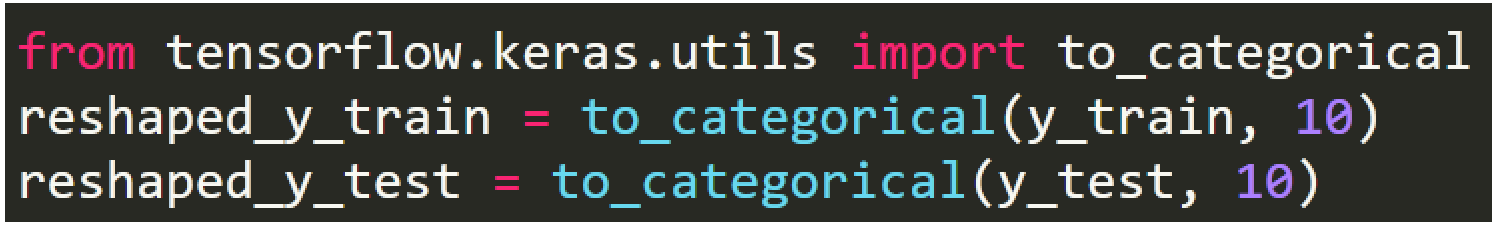


label

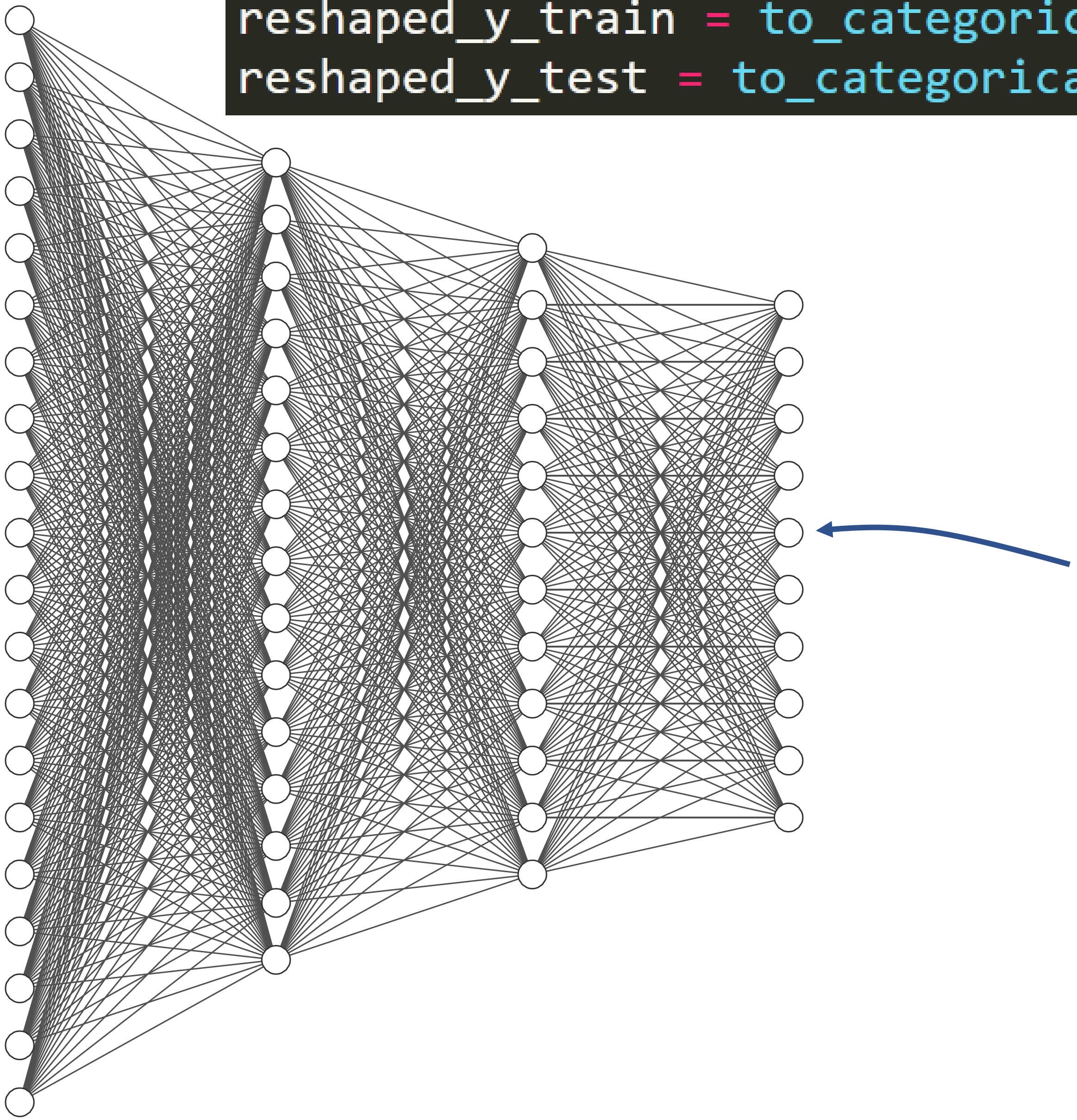
5, 0, 4, 1, 9, 2, 1, 3,
1, 4, 3, 5, 3, 6, 1, 7,
2, 8, 6, 9, 4, 0, 9, 1,
1, 2, 4, 3, 2, 7, …

0	**=**	**[1 0 0 0 0 0 0 0 0 0]**
1	**=**	**[0 1 0 0 0 0 0 0 0 0]**
2	**=**	**[0 0 1 0 0 0 0 0 0 0]**
3	**=**	**[0 0 0 1 0 0 0 0 0 0]**
4	**=**	**[0 0 0 0 1 0 0 0 0 0]**
5	**=**	**[0 0 0 0 0 1 0 0 0 0]**
6	**=**	**[0 0 0 0 0 0 1 0 0 0]**
7	**=**	**[0 0 0 0 0 0 0 1 0 0]**
8	**=**	**[0 0 0 0 0 0 0 0 1 0]**
9	**=**	**[0 0 0 0 0 0 0 0 0 1]**

kita sebut tahapan ini dengan encoding



```python
from tensorflow.keras.utils import to_categorical
reshaped_y_train = to_categorical(y_train, 10)
reshaped_y_test = to_categorical(y_test, 10)
```

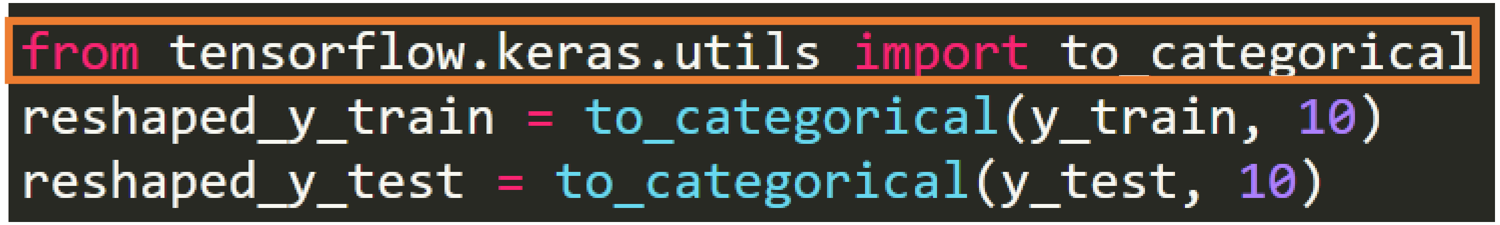


label

5, 0, 4, 1, 9, 2, 1, 3,
1, 4, 3, 5, 3, 6, 1, 7,
2, 8, 6, 9, 4, 0, 9, 1,
1, 2, 4, 3, 2, 7, …

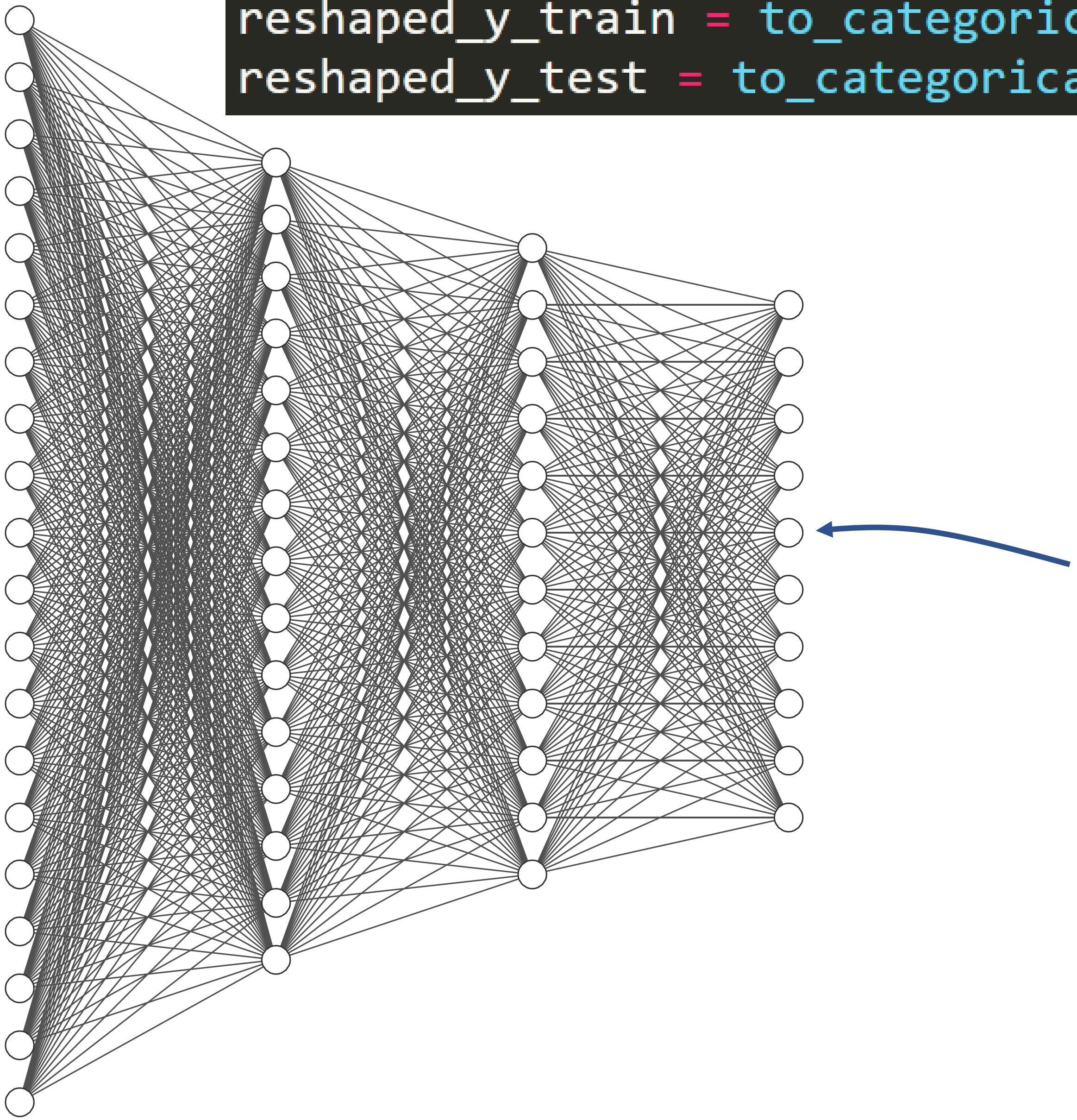
0 = [1 0 0 0 0 0 0 0 0 0]
1 = [0 1 0 0 0 0 0 0 0 0]
2 = [0 0 1 0 0 0 0 0 0 0]
3 = [0 0 0 1 0 0 0 0 0 0]
4 = [0 0 0 0 1 0 0 0 0 0]
5 = [0 0 0 0 0 1 0 0 0 0]
6 = [0 0 0 0 0 0 1 0 0 0]
7 = [0 0 0 0 0 0 0 1 0 0]
8 = [0 0 0 0 0 0 0 0 1 0]
9 = [0 0 0 0 0 0 0 0 0 1]

kita sebut tahapan ini dengan encoding



```
from tensorflow.keras.utils import to_categorical
reshaped_y_train = to_categorical(y_train, 10)
reshaped_y_test = to_categorical(y_test, 10)
```

import modul untuk encoding

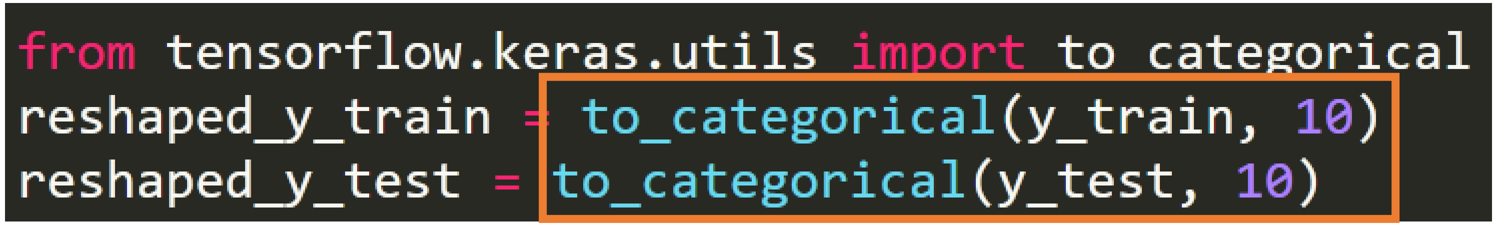


label

5, 0, 4, 1, 9, 2, 1, 3,
1, 4, 3, 5, 3, 6, 1, 7,
2, 8, 6, 9, 4, 0, 9, 1,
1, 2, 4, 3, 2, 7, …

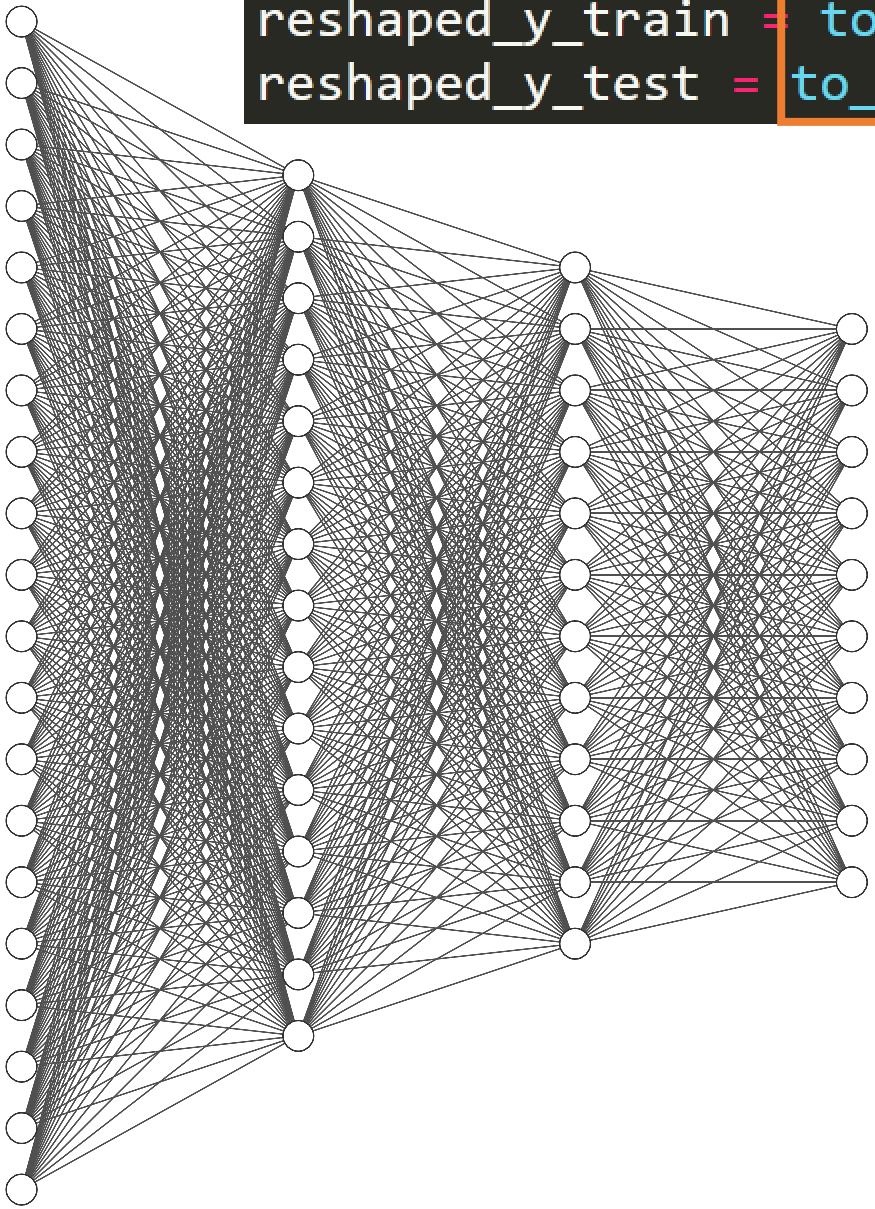
0 = [1 0 0 0 0 0 0 0 0 0]
1 = [0 1 0 0 0 0 0 0 0 0]
2 = [0 0 1 0 0 0 0 0 0 0]
3 = [0 0 0 1 0 0 0 0 0 0]
4 = [0 0 0 0 1 0 0 0 0 0]
5 = [0 0 0 0 0 1 0 0 0 0]
6 = [0 0 0 0 0 0 1 0 0 0]
7 = [0 0 0 0 0 0 0 1 0 0]
8 = [0 0 0 0 0 0 0 0 1 0]
9 = [0 0 0 0 0 0 0 0 0 1]

kita sebut tahapan ini dengan encoding



```
from tensorflow.keras.utils import to_categorical
reshaped_y_train = to_categorical(y_train, 10)
reshaped_y_test = to_categorical(y_test, 10)
```

encode label y_train dan y_test

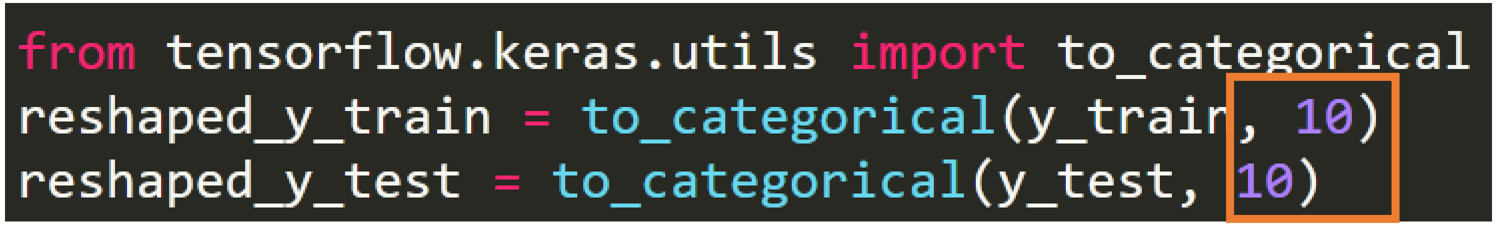


label

5, 0, 4, 1, 9, 2, 1, 3, 1, 4, 3, 5, 3, 6, 1, 7, 2, 8, 6, 9, 4, 0, 9, 1, 1, 2, 4, 3, 2, 7, …

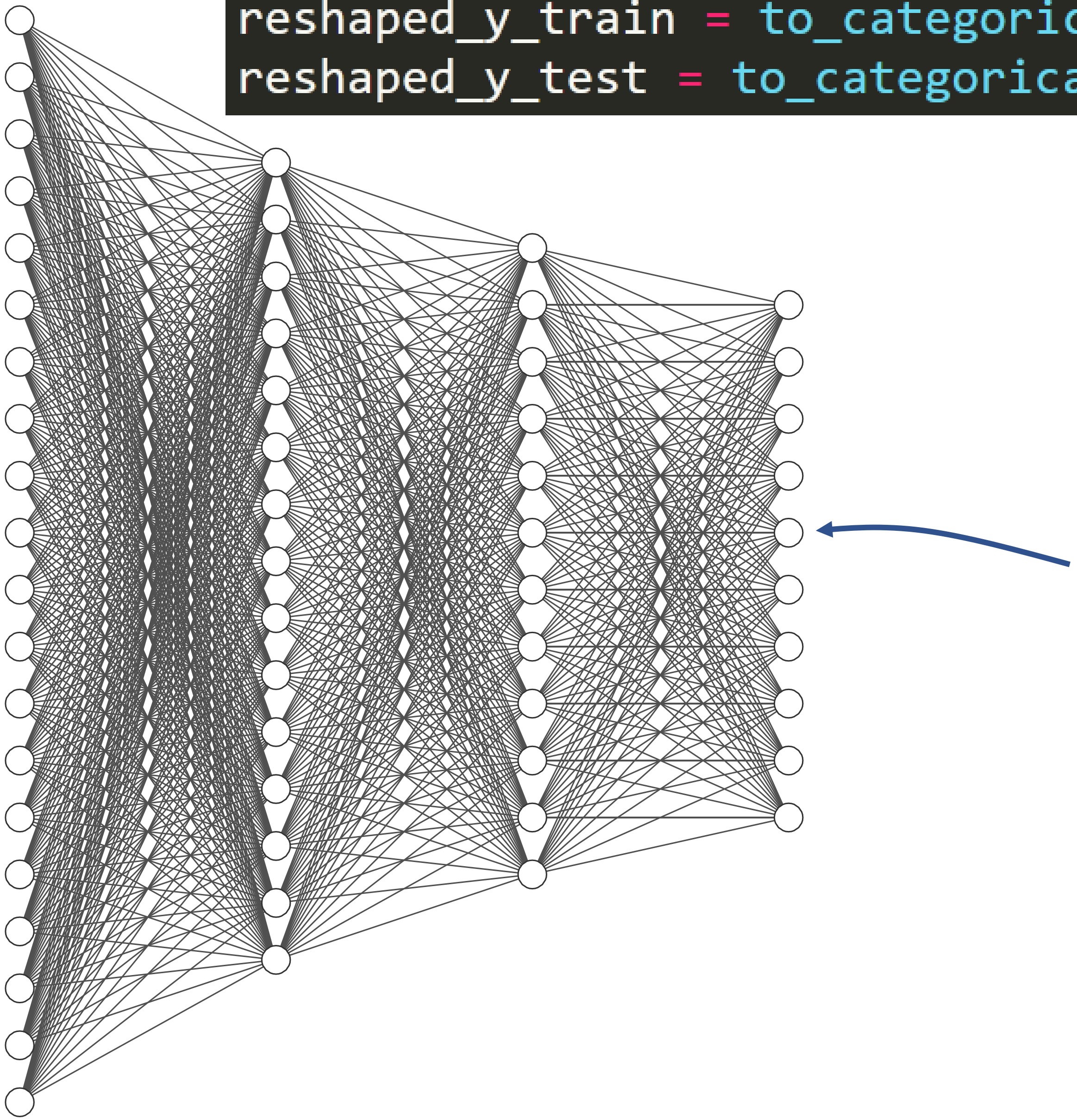
0 = [1 0 0 0 0 0 0 0 0 0]
1 = [0 1 0 0 0 0 0 0 0 0]
2 = [0 0 1 0 0 0 0 0 0 0]
3 = [0 0 0 1 0 0 0 0 0 0]
4 = [0 0 0 0 1 0 0 0 0 0]
5 = [0 0 0 0 0 1 0 0 0 0]
6 = [0 0 0 0 0 0 1 0 0 0]
7 = [0 0 0 0 0 0 0 1 0 0]
8 = [0 0 0 0 0 0 0 0 1 0]
9 = [0 0 0 0 0 0 0 0 0 1]

kita sebut tahapan ini dengan encoding



```
from tensorflow.keras.utils import to_categorical
reshaped_y_train = to_categorical(y_train, 10)
reshaped_y_test = to_categorical(y_test, 10)
```

10 merupakan jumlah kelas label (0-9)



label

5, 0, 4, 1, 9, 2, 1, 3,
1, 4, 3, 5, 3, 6, 1, 7,
2, 8, 6, 9, 4, 0, 9, 1,
1, 2, 4, 3, 2, 7, …

0 = [1 0 0 0 0 0 0 0 0 0]
1 = [0 1 0 0 0 0 0 0 0 0]
2 = [0 0 1 0 0 0 0 0 0 0]
3 = [0 0 0 1 0 0 0 0 0 0]
4 = [0 0 0 0 1 0 0 0 0 0]
5 = [0 0 0 0 0 1 0 0 0 0]
6 = [0 0 0 0 0 0 1 0 0 0]
7 = [0 0 0 0 0 0 0 1 0 0]
8 = [0 0 0 0 0 0 0 0 1 0]
9 = [0 0 0 0 0 0 0 0 0 1]

kita sebut tahapan ini dengan encoding

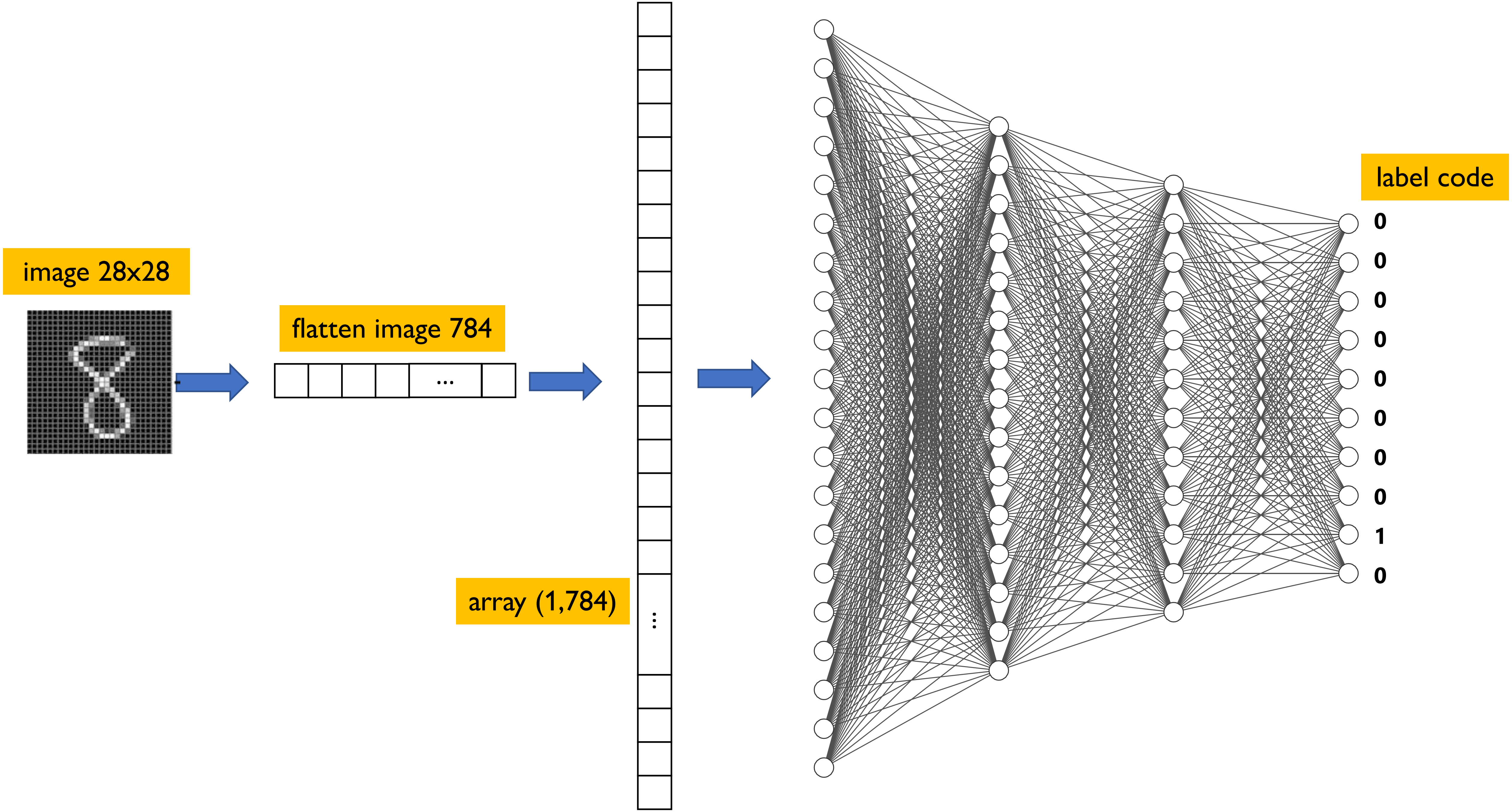
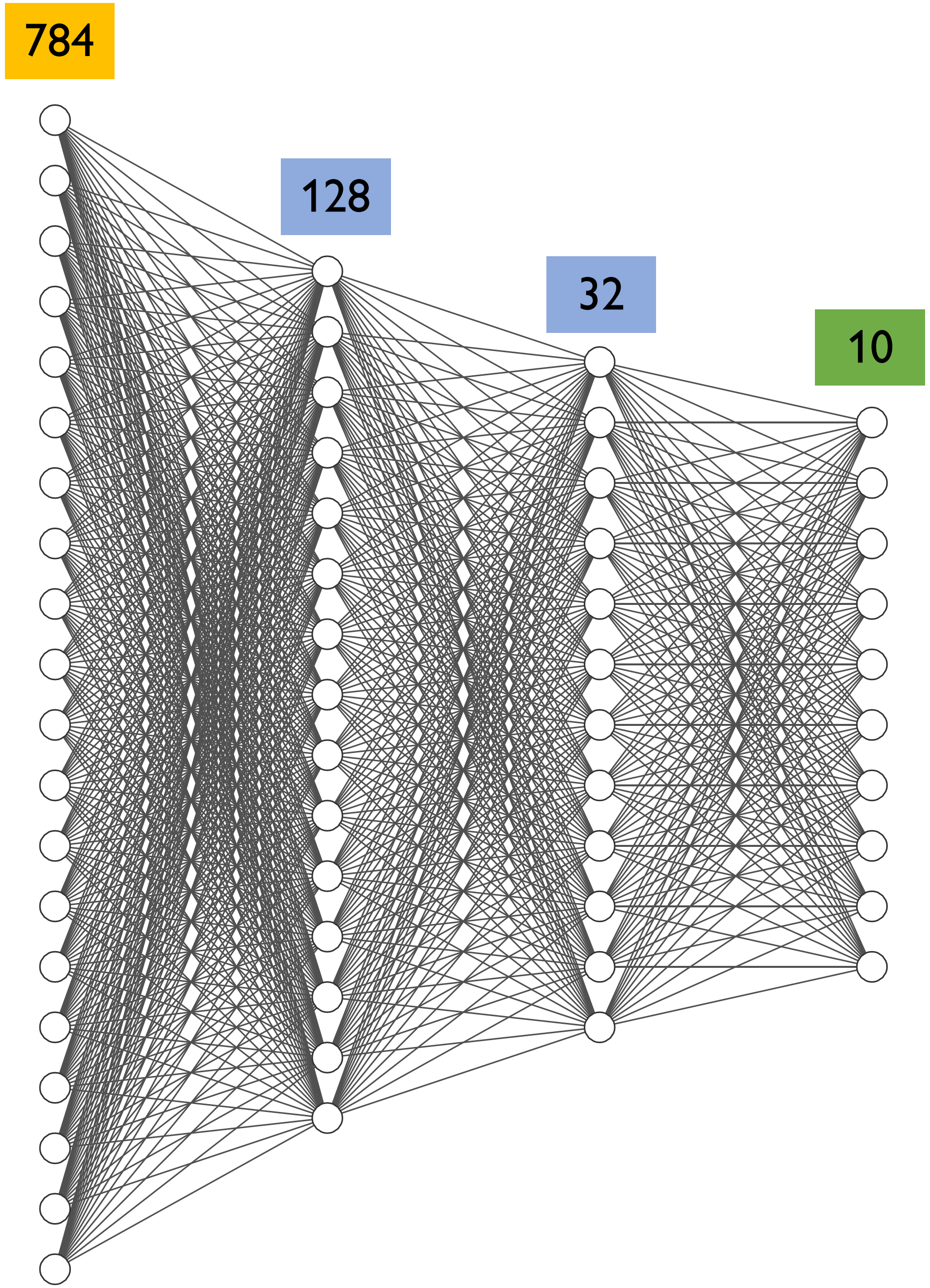


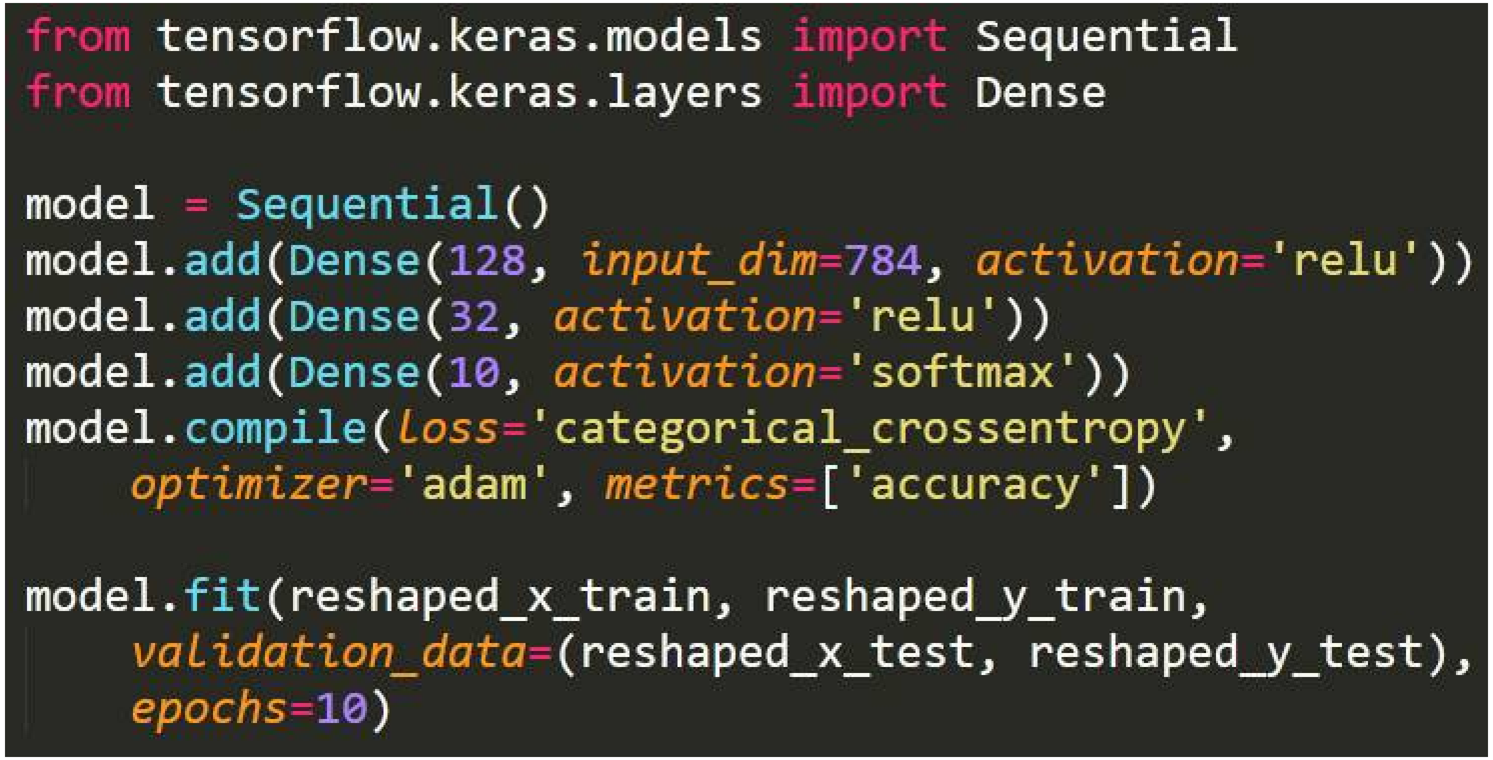
image 28x28

flatten image 784

array (1,784)

label code



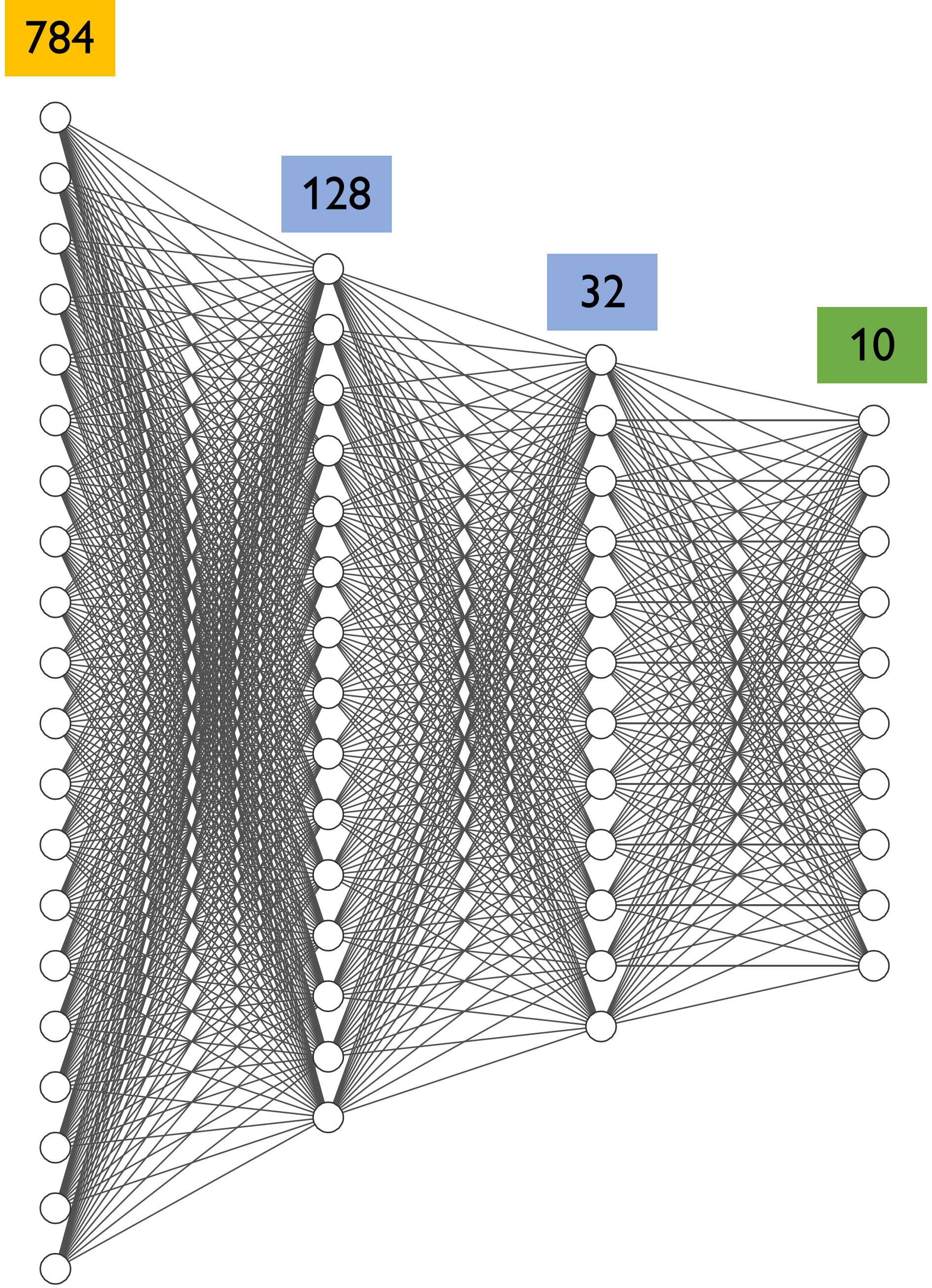


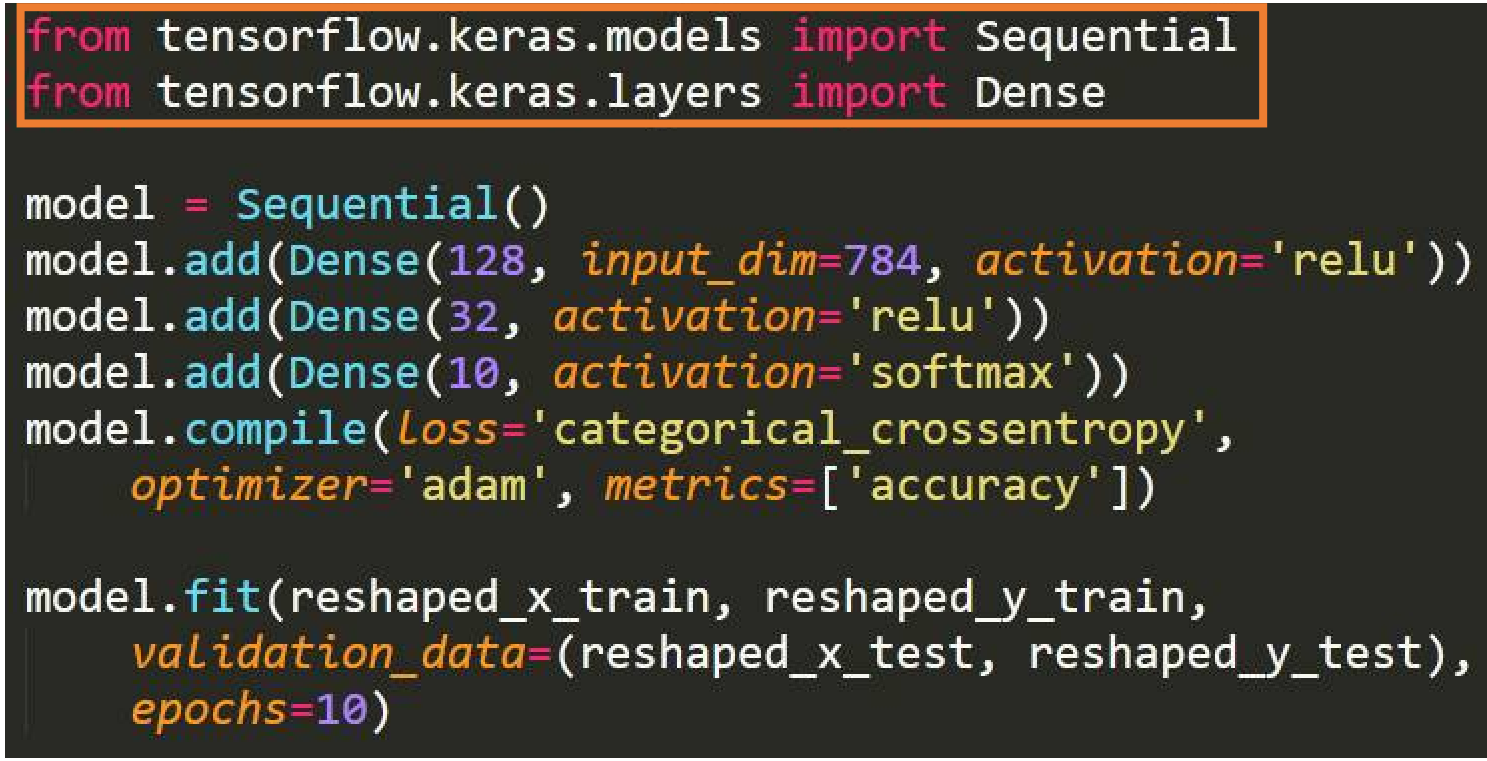
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

input layer

hidden layer

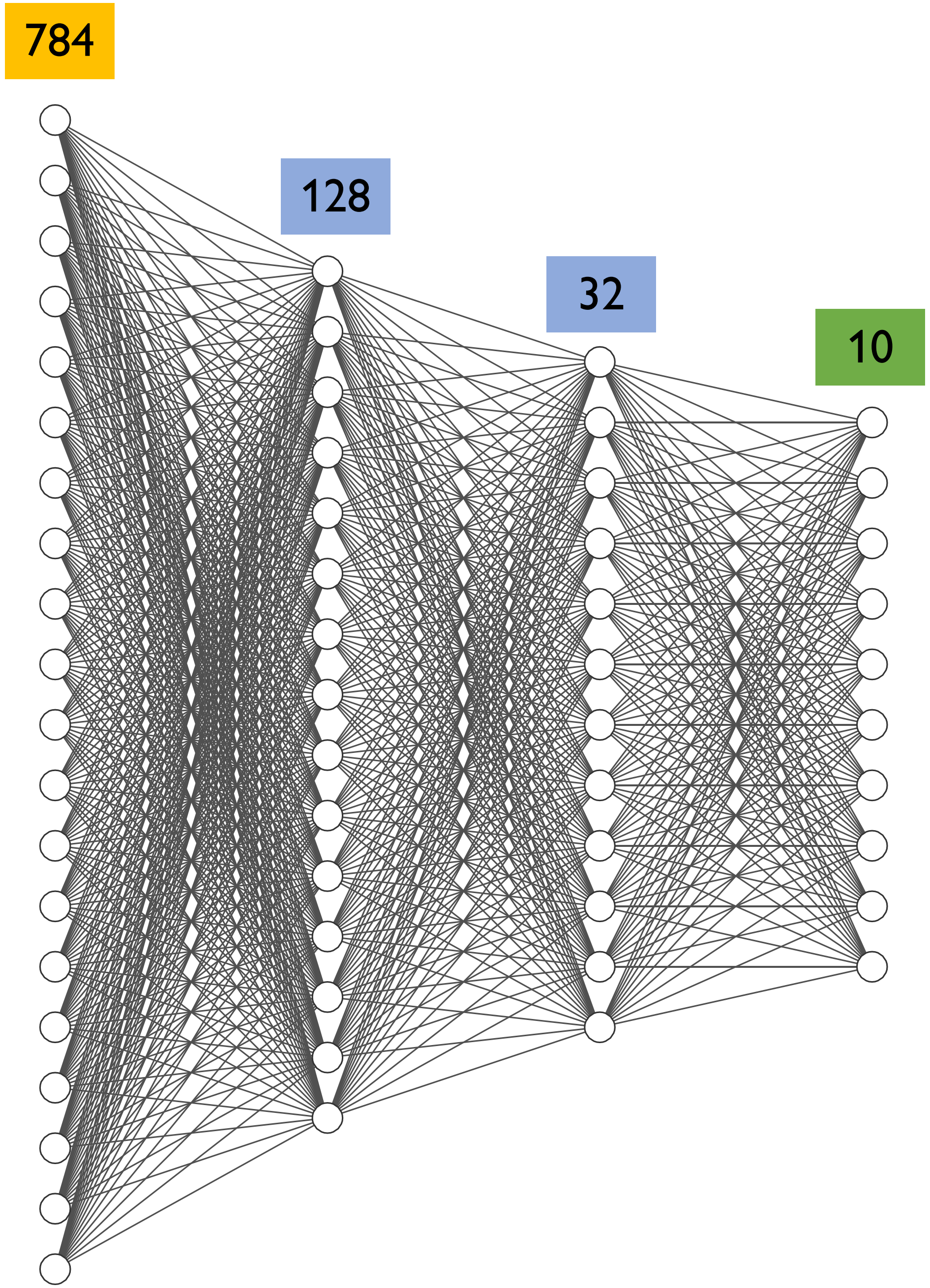
output layer





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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

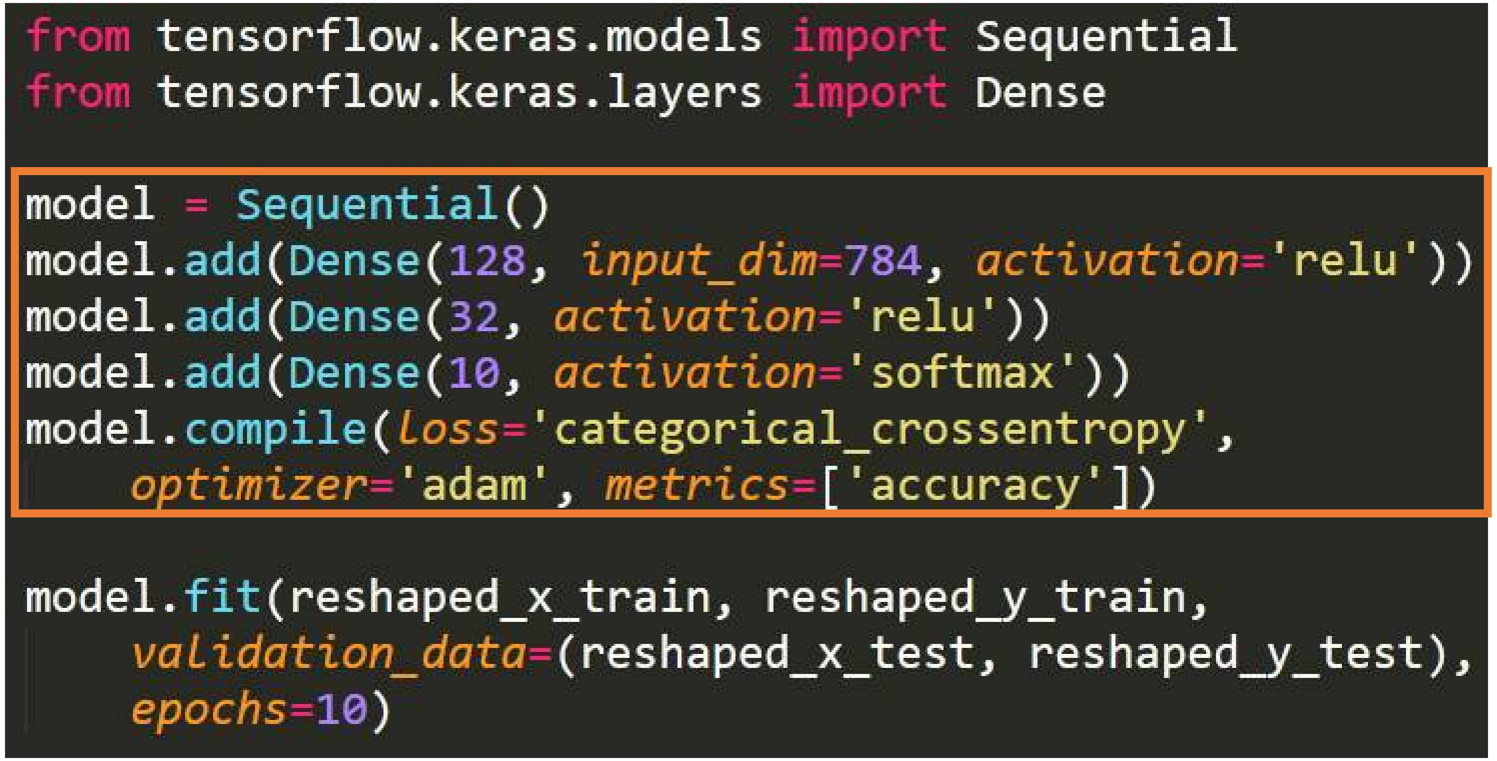
note :
import modules



input layer

hidden layer

output layer



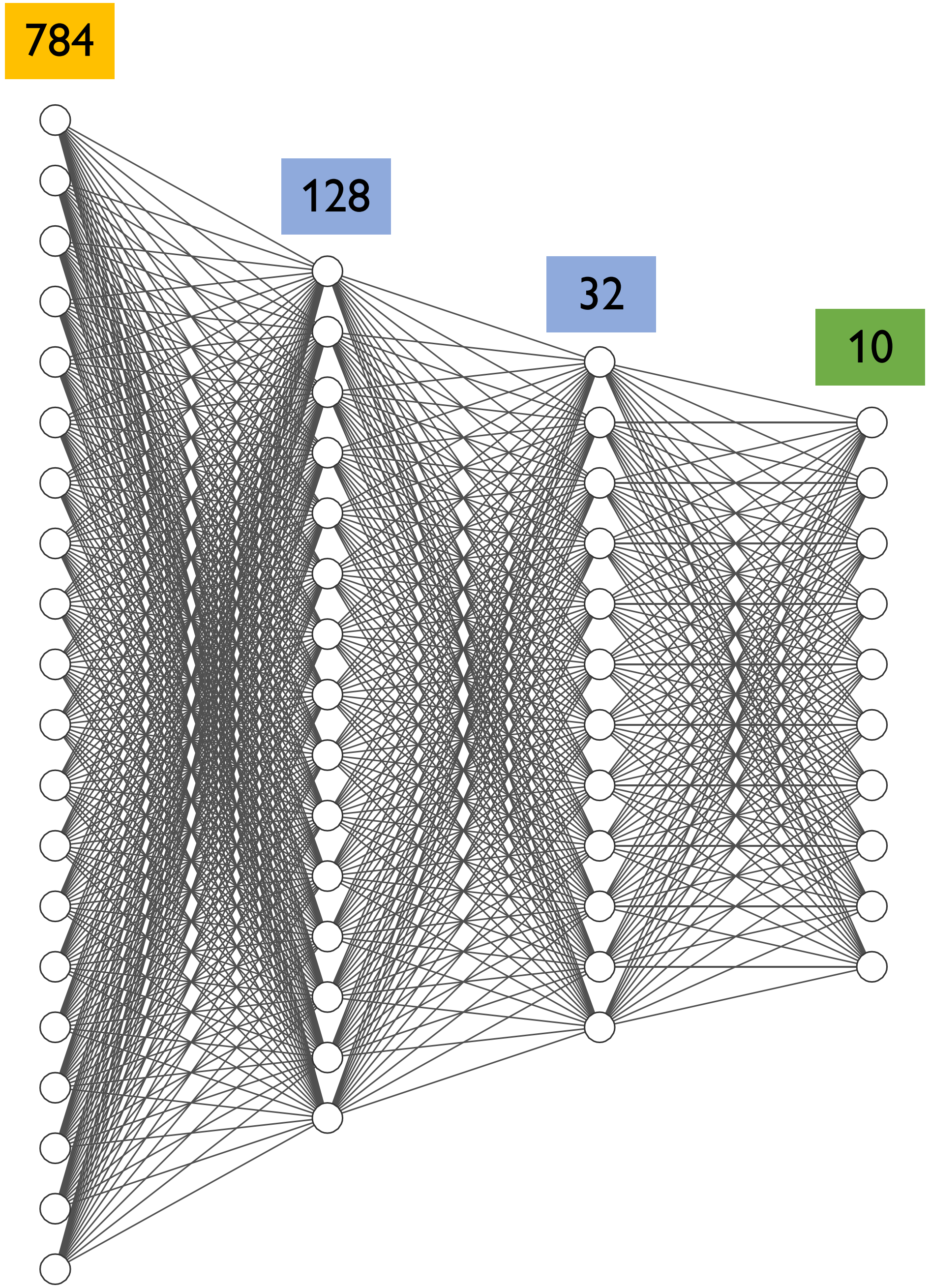
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

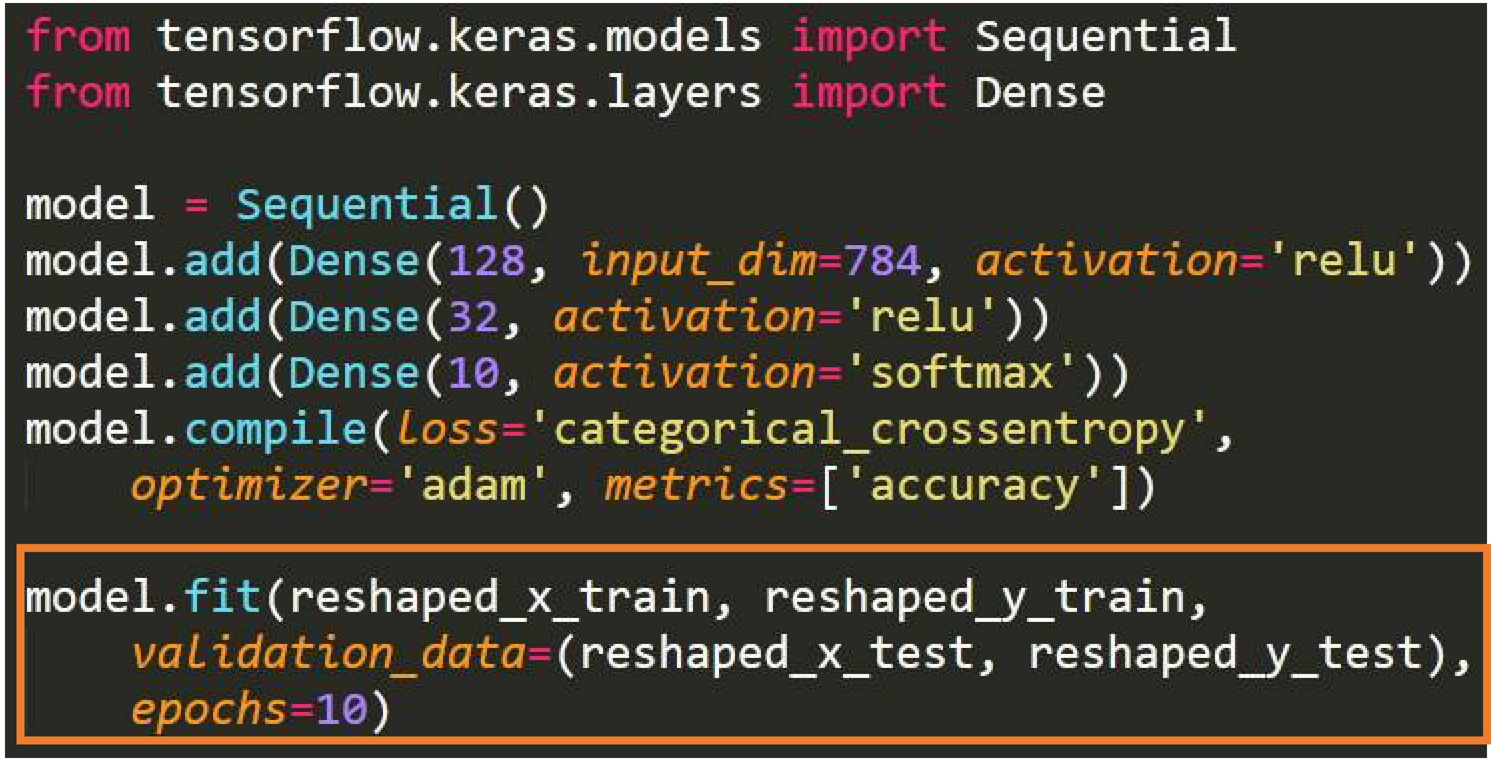
note :
buat arsitektur/model ANN

input layer

hidden layer

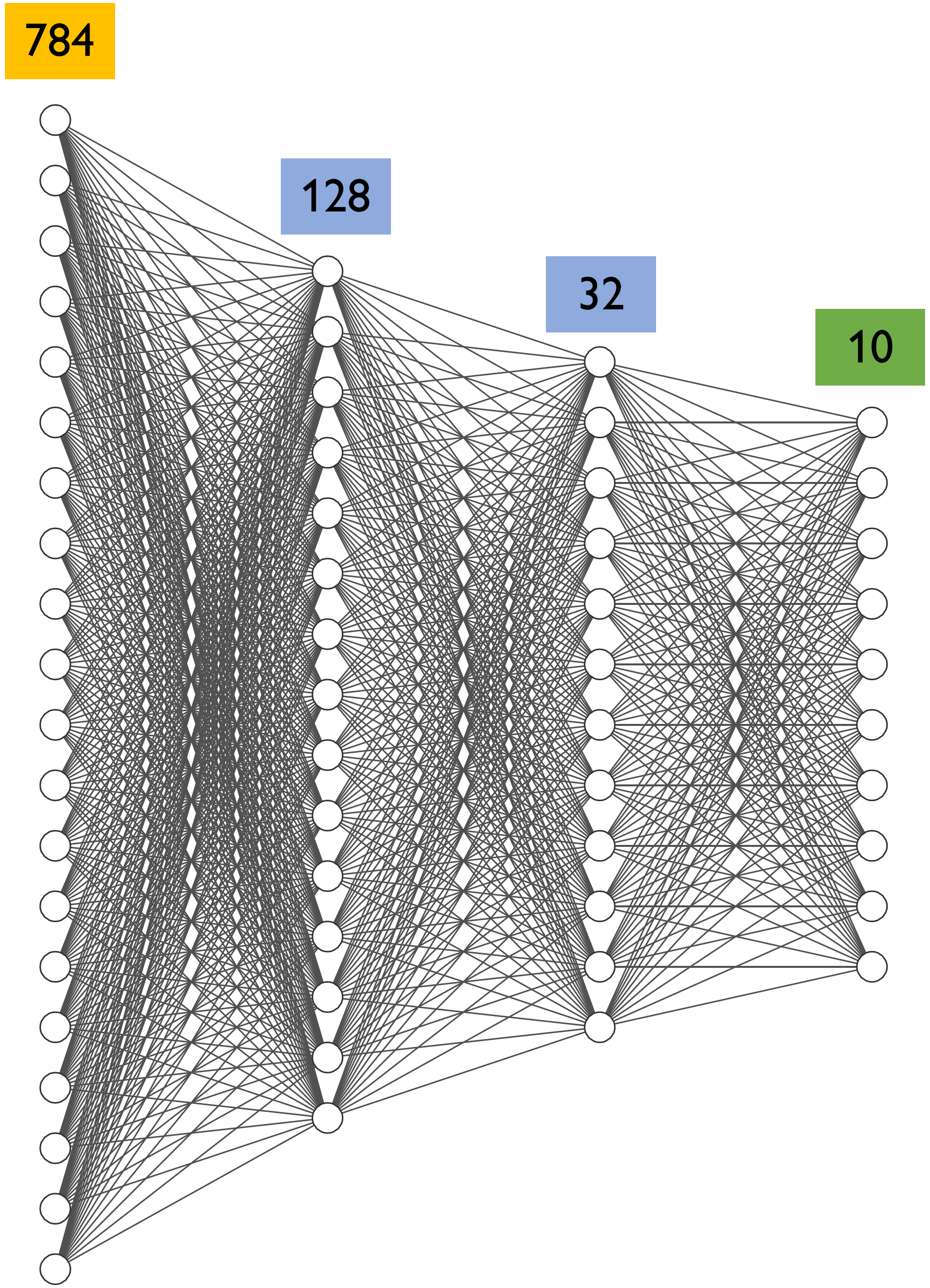
output layer





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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

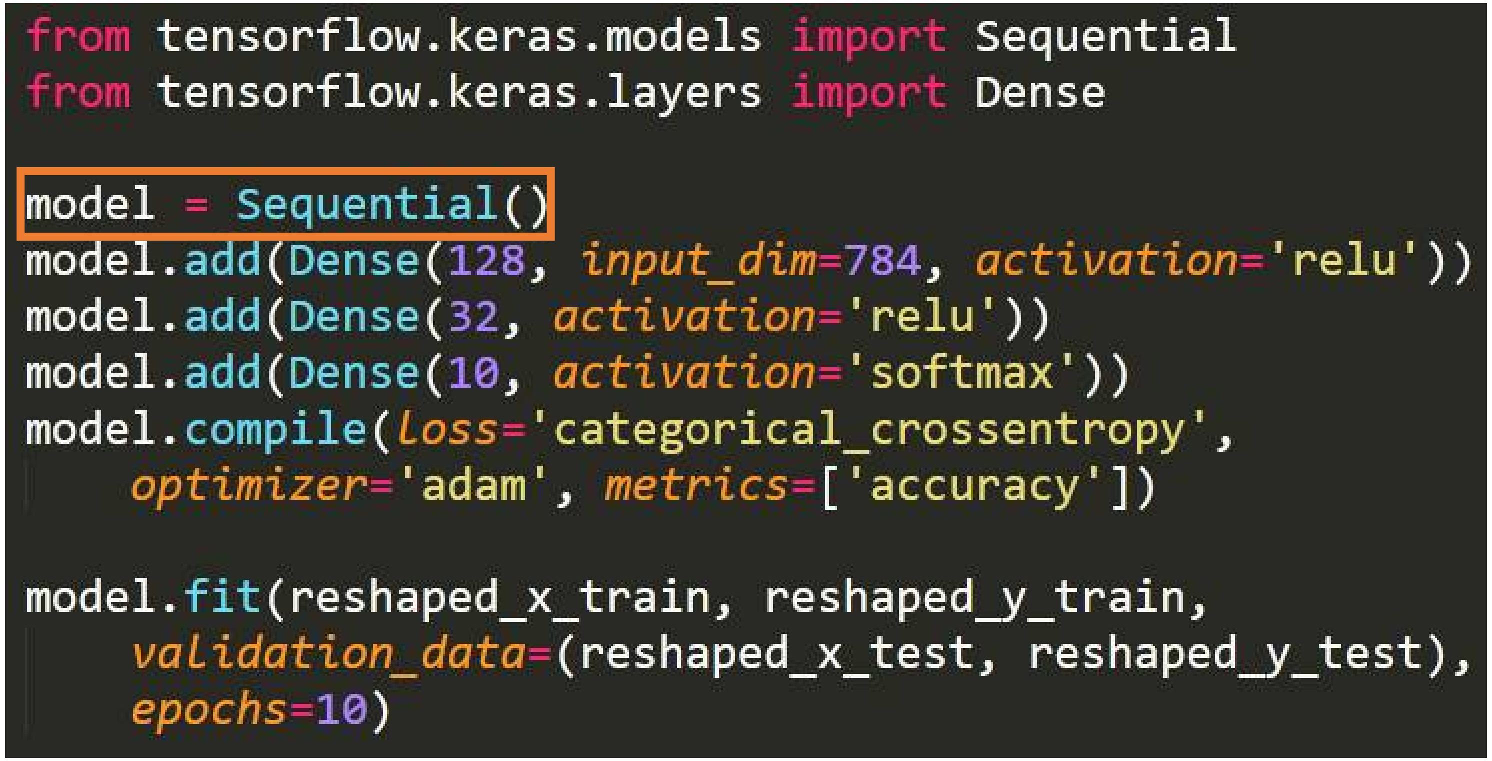
note :
train/latih model



🟨 input layer

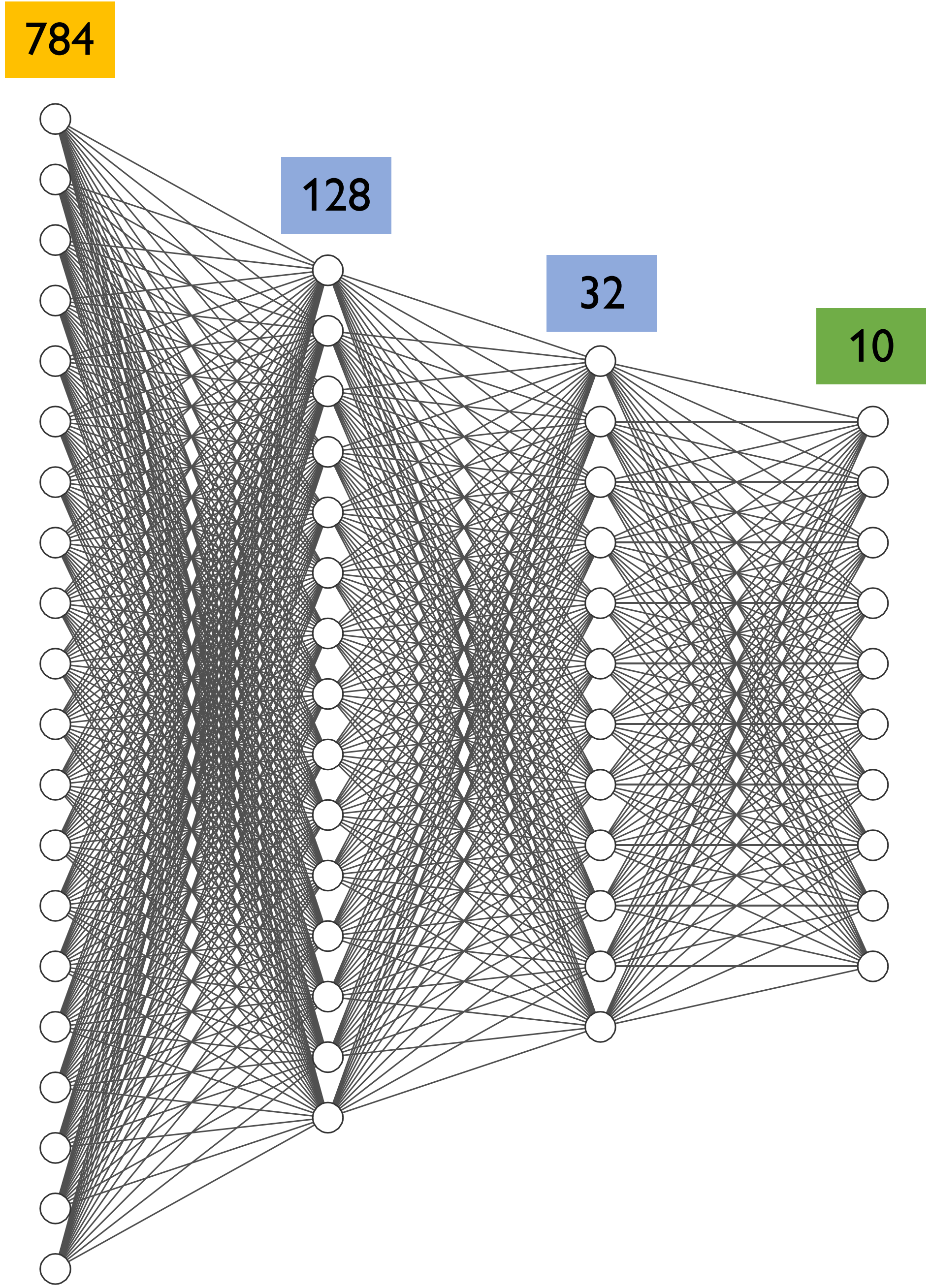
🟦 hidden layer

🟩 output layer



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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

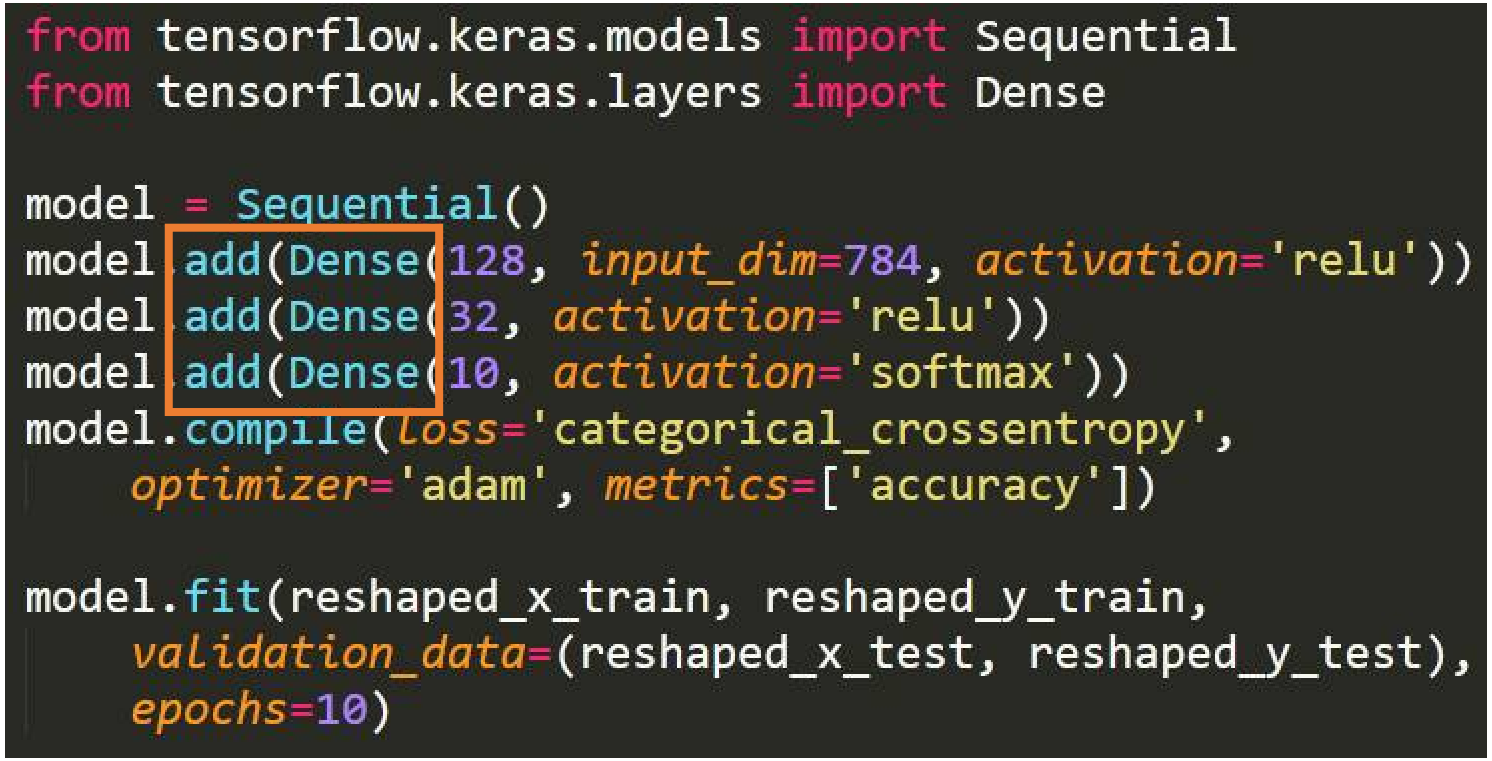
note : **Sequential()**
buat model dengan tumpukan layer yang dimulai dari input hingga output



input layer

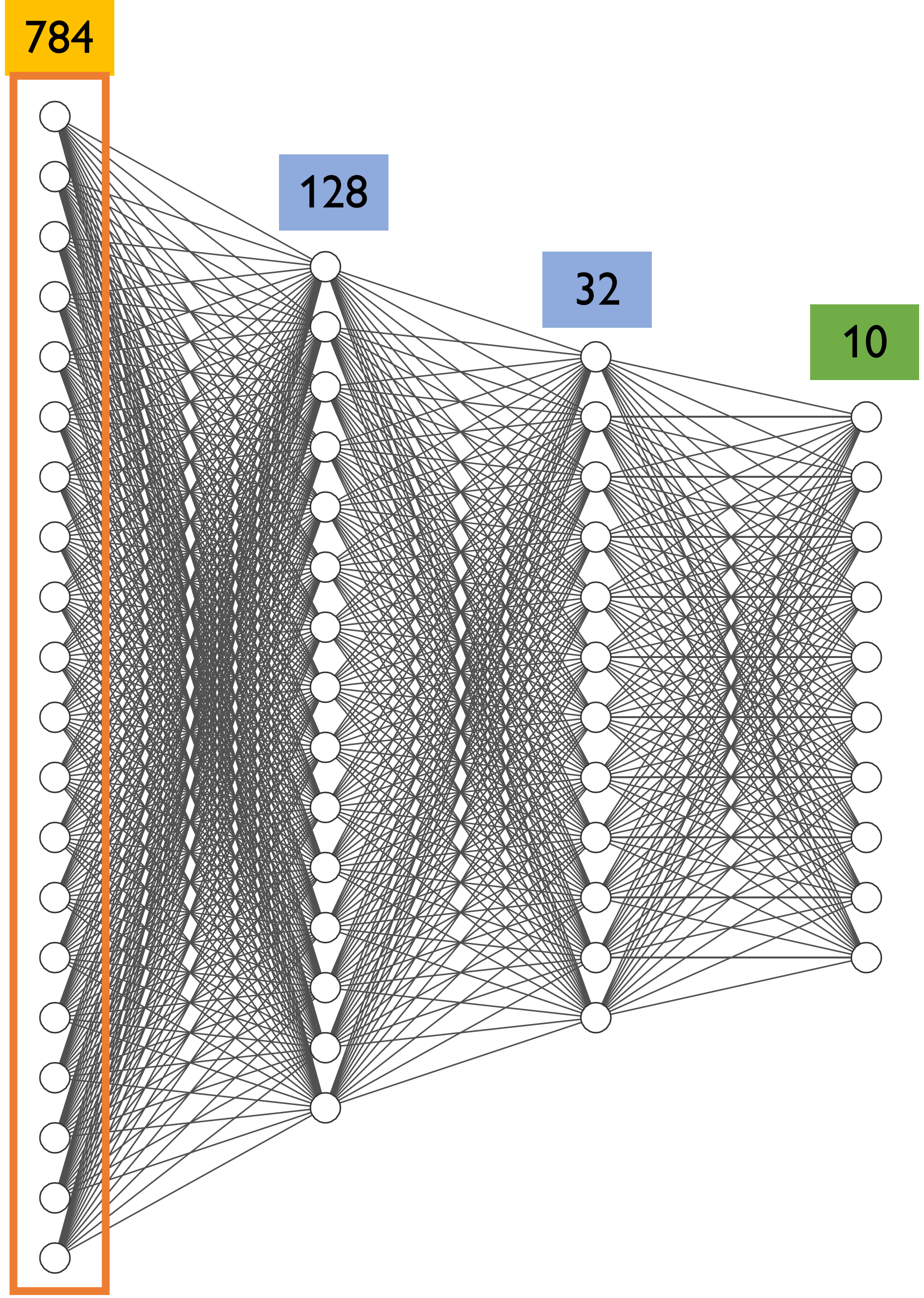
hidden layer

output layer



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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

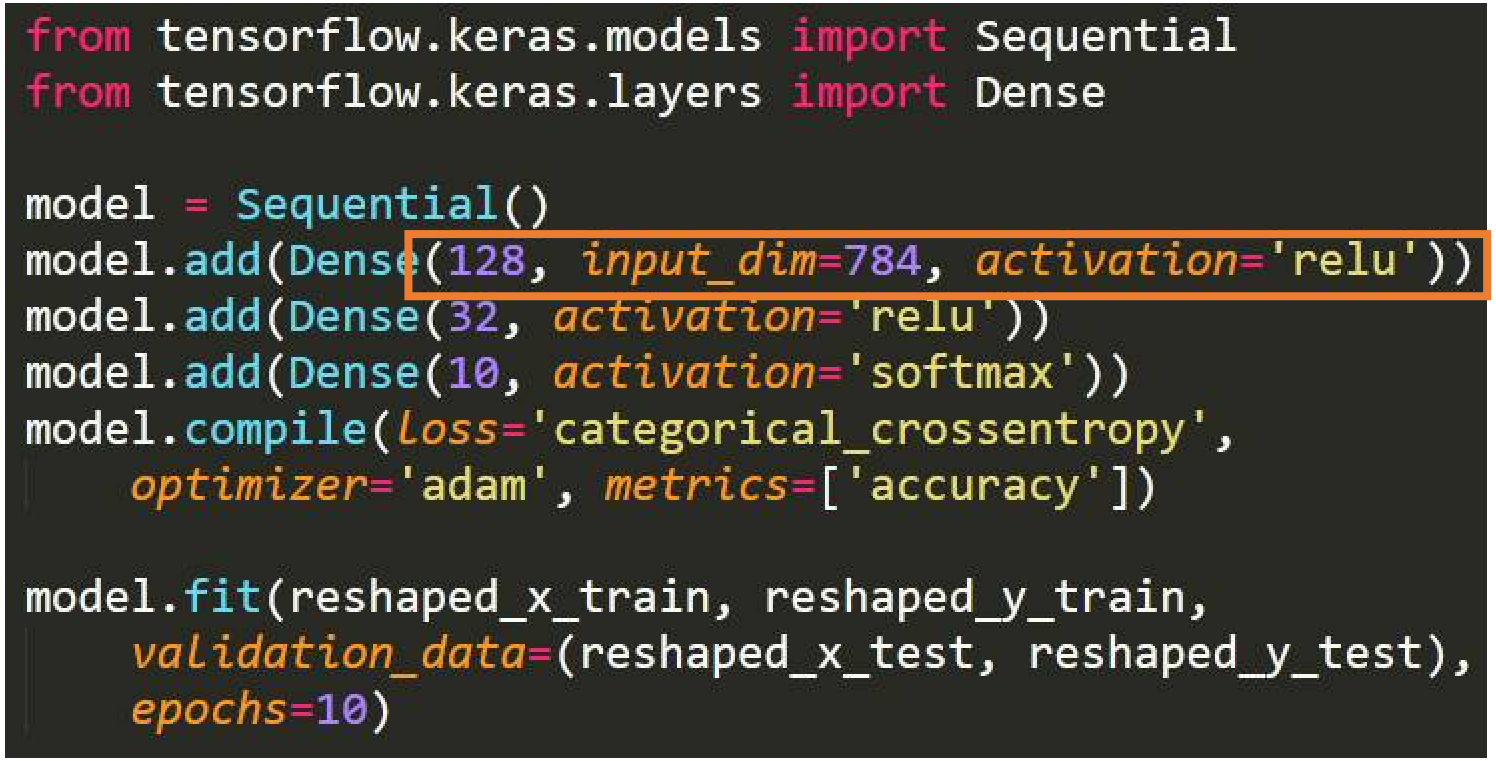
note : **.add(Dense())**
function untuk menambahkan layer



input layer

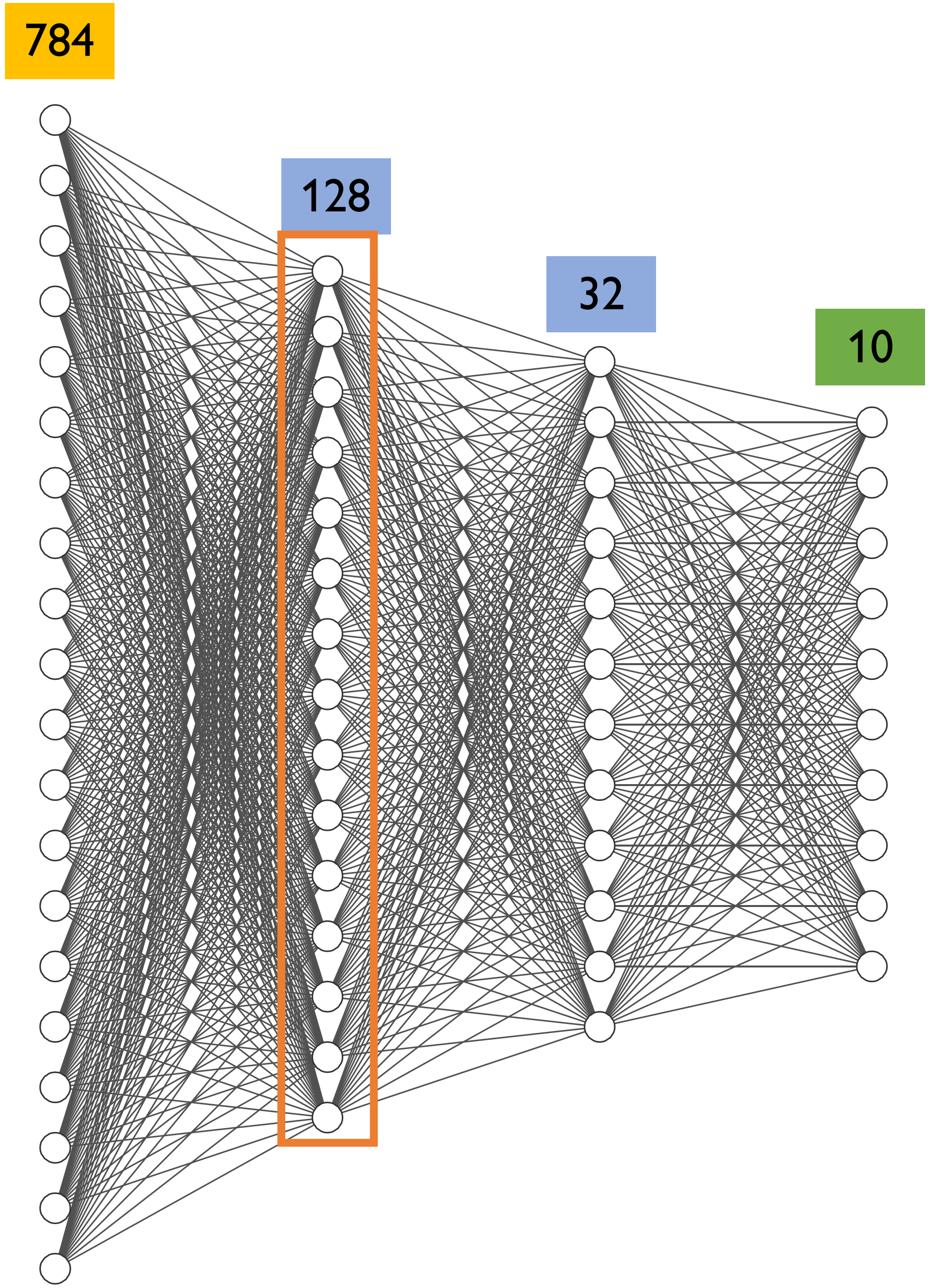
hidden layer

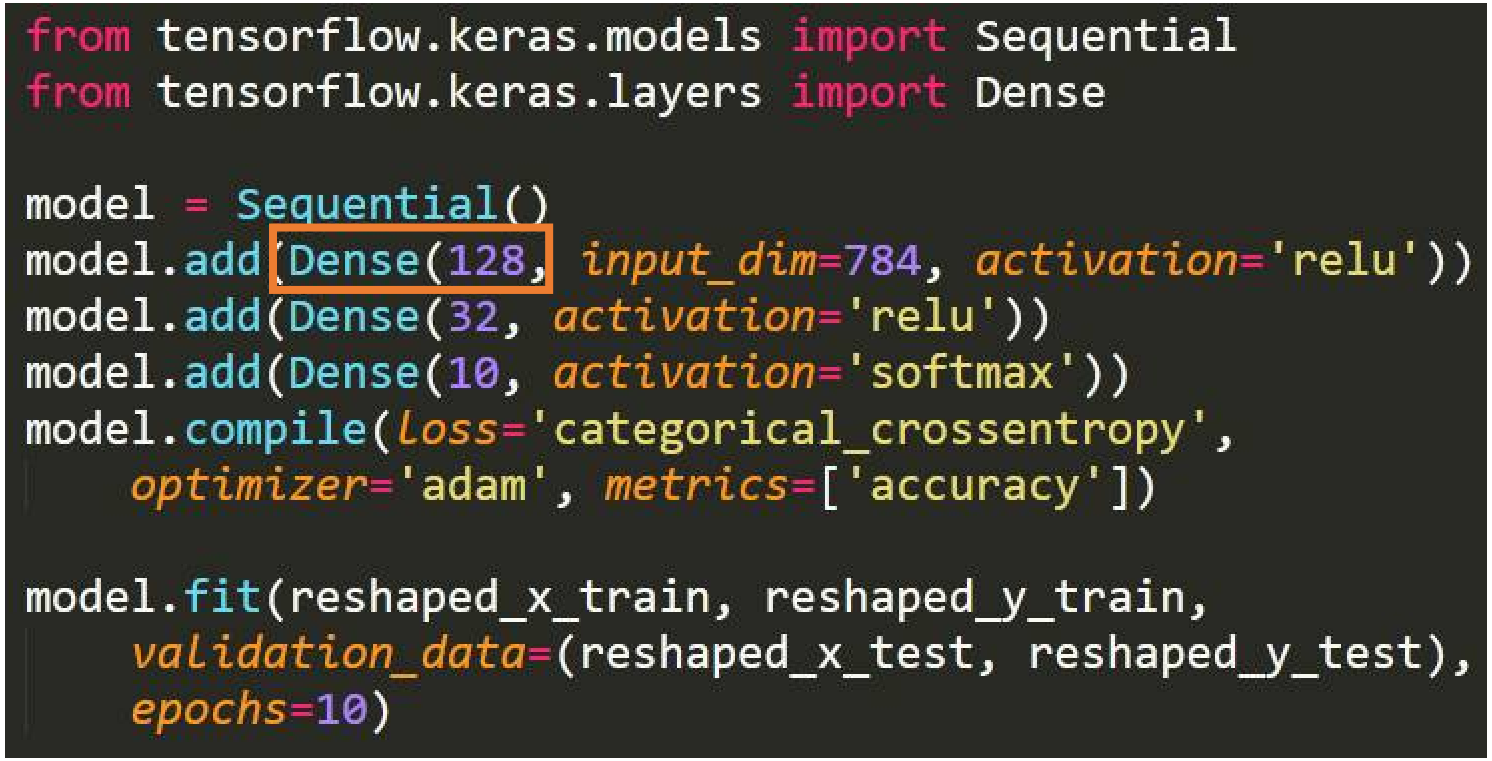
output layer



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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

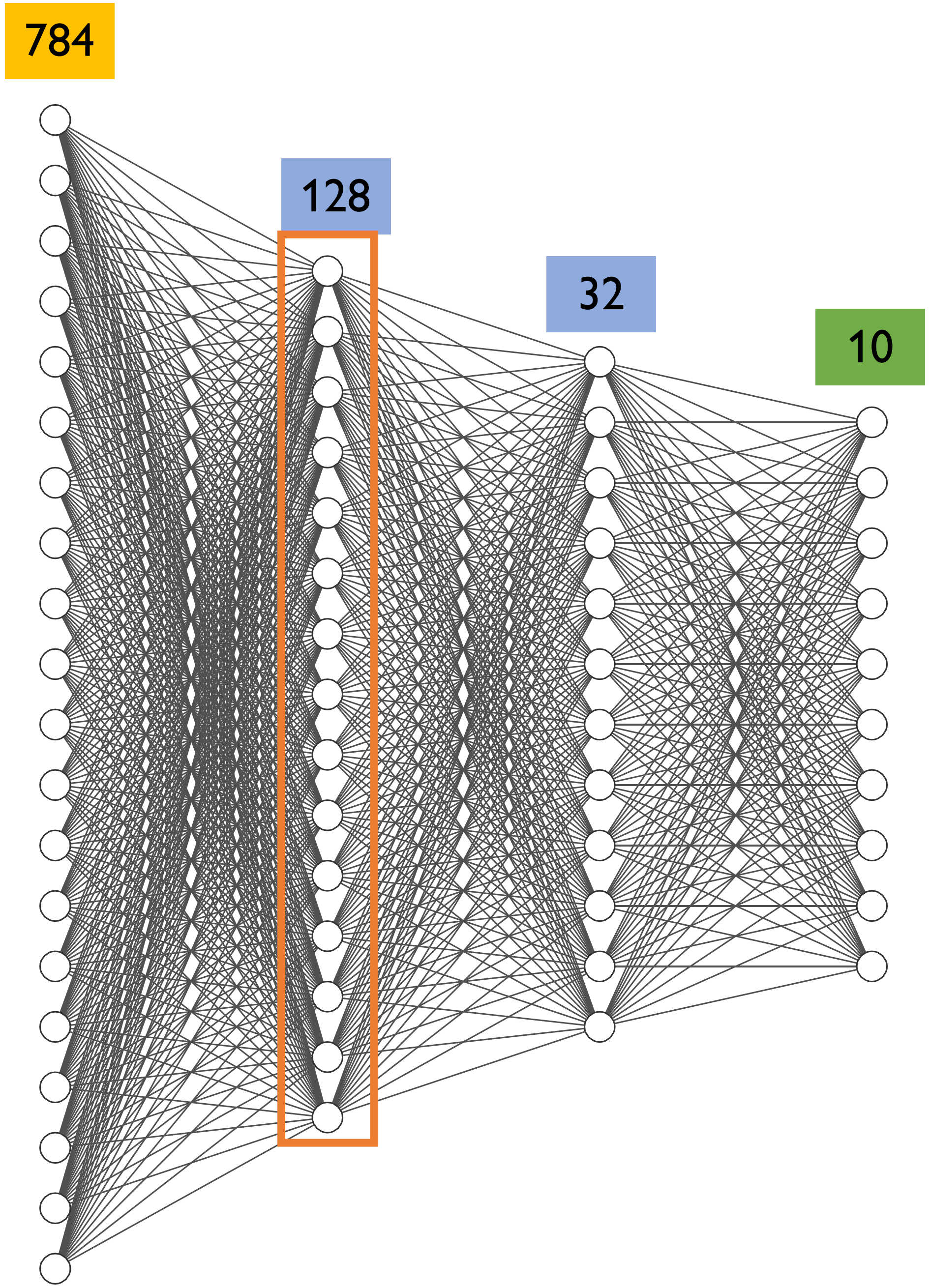
note :
buat layer input dengan node sebanyak **784 unit**, output hasil layer akan dihitung dengan fungsi aktivasi ReLU (Rectifier Linear Unit). fungsi aktivasi berguna untuk mengenalkan non-linearitas pada model





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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

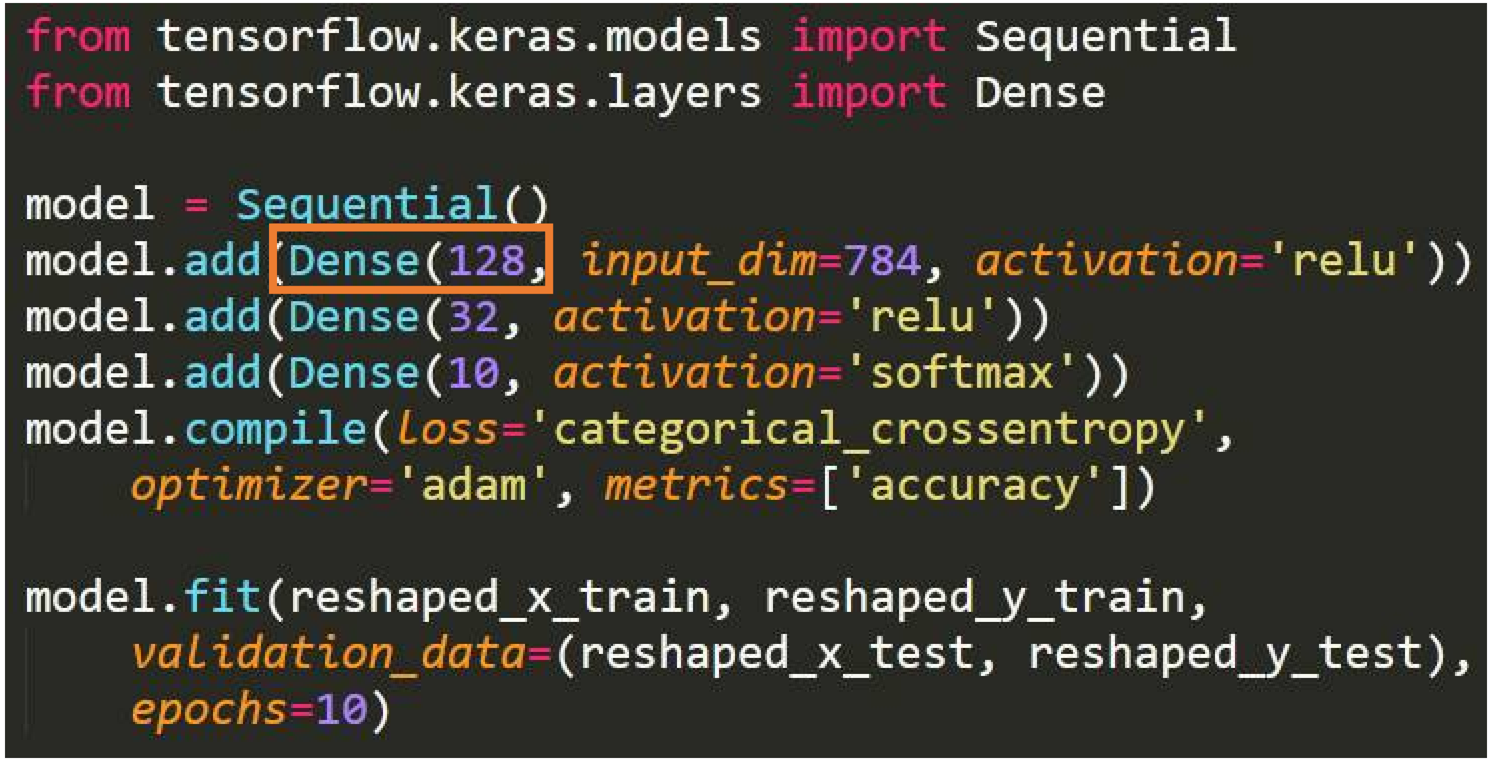
note :
khusus untuk layer input, kita dapat mendeklarasikan dua buah layer sekaligus. code yang ditandai adalah layer berikutnya setelah layer input. kita akan membuat layer dengan **128 node**



input layer

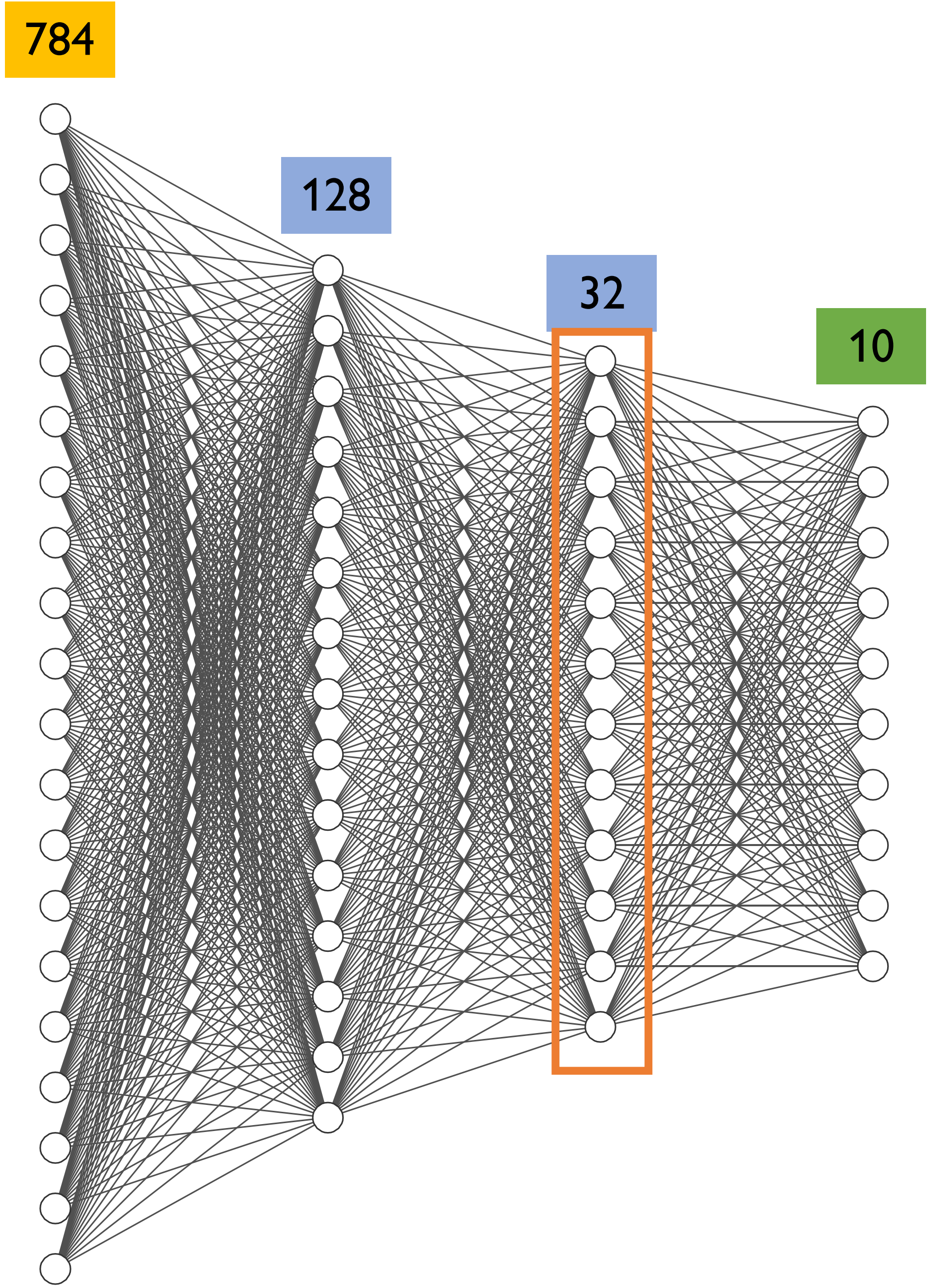
hidden layer

output layer



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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

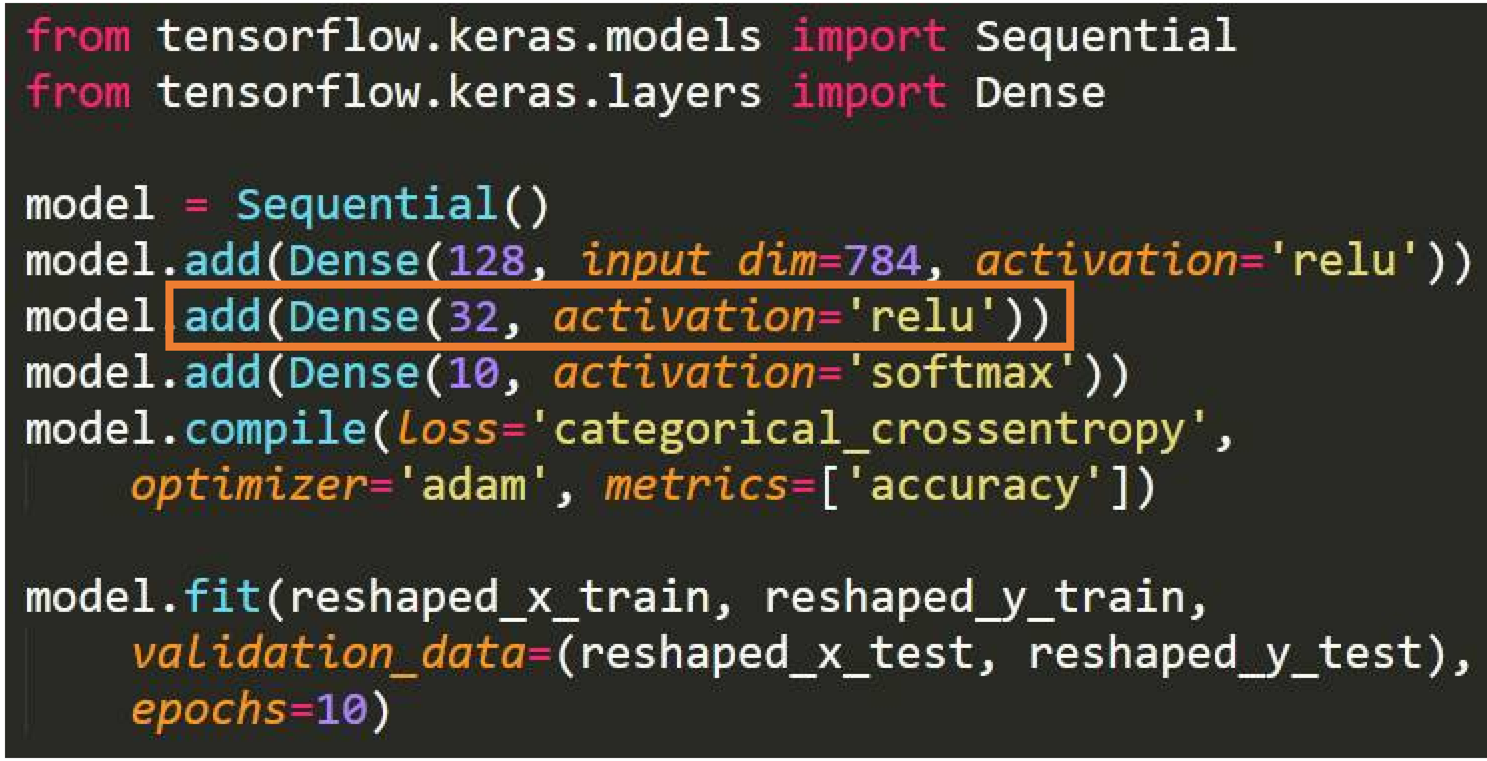
note :
khusus untuk layer input, kita dapat mendeklarasikan dua buah layer sekaligus. code yang ditandai adalah layer berikutnya setelah layer input. kita akan membuat layer dengan **128 node**



input layer

hidden layer

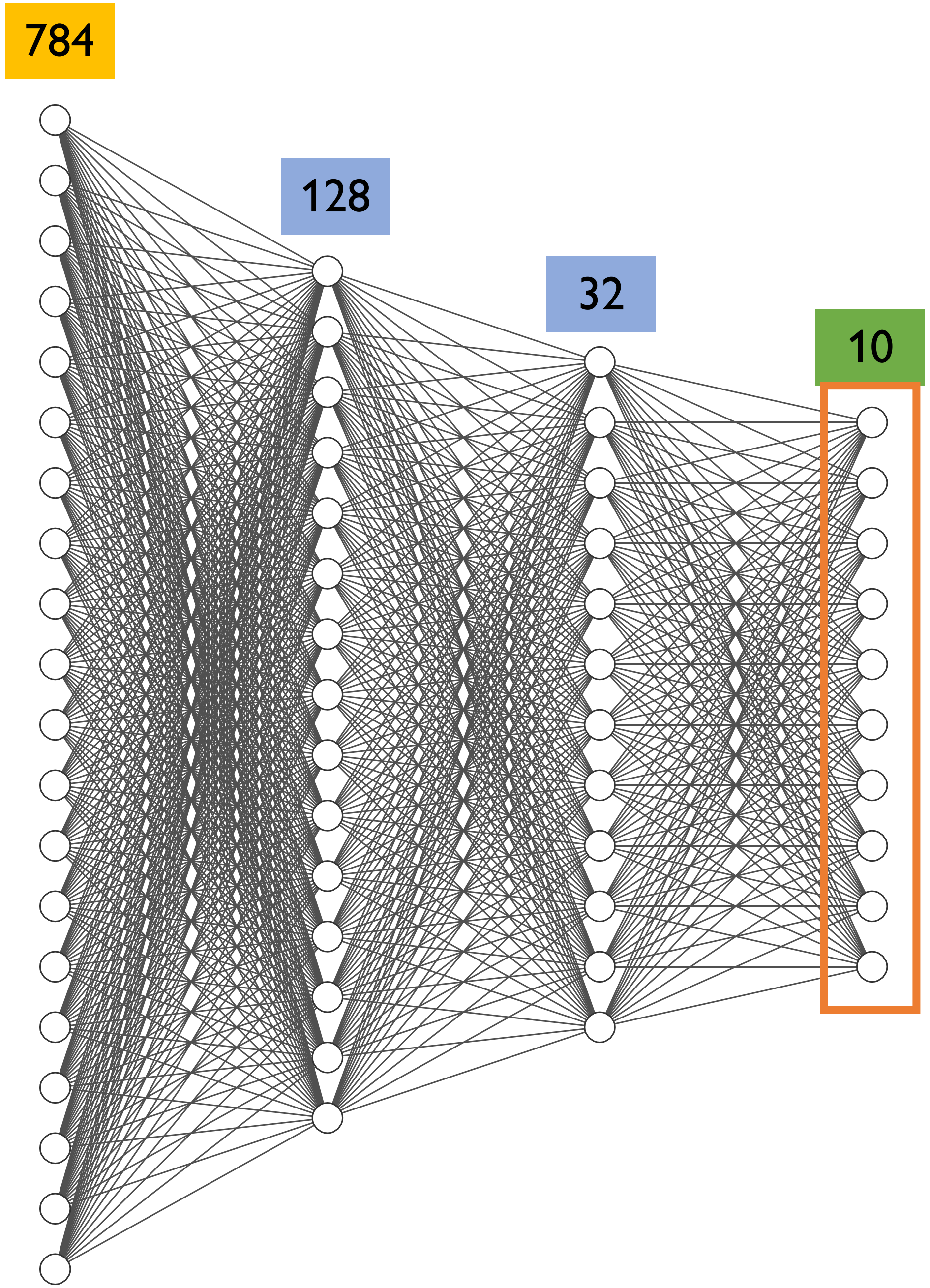
output layer



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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

note :

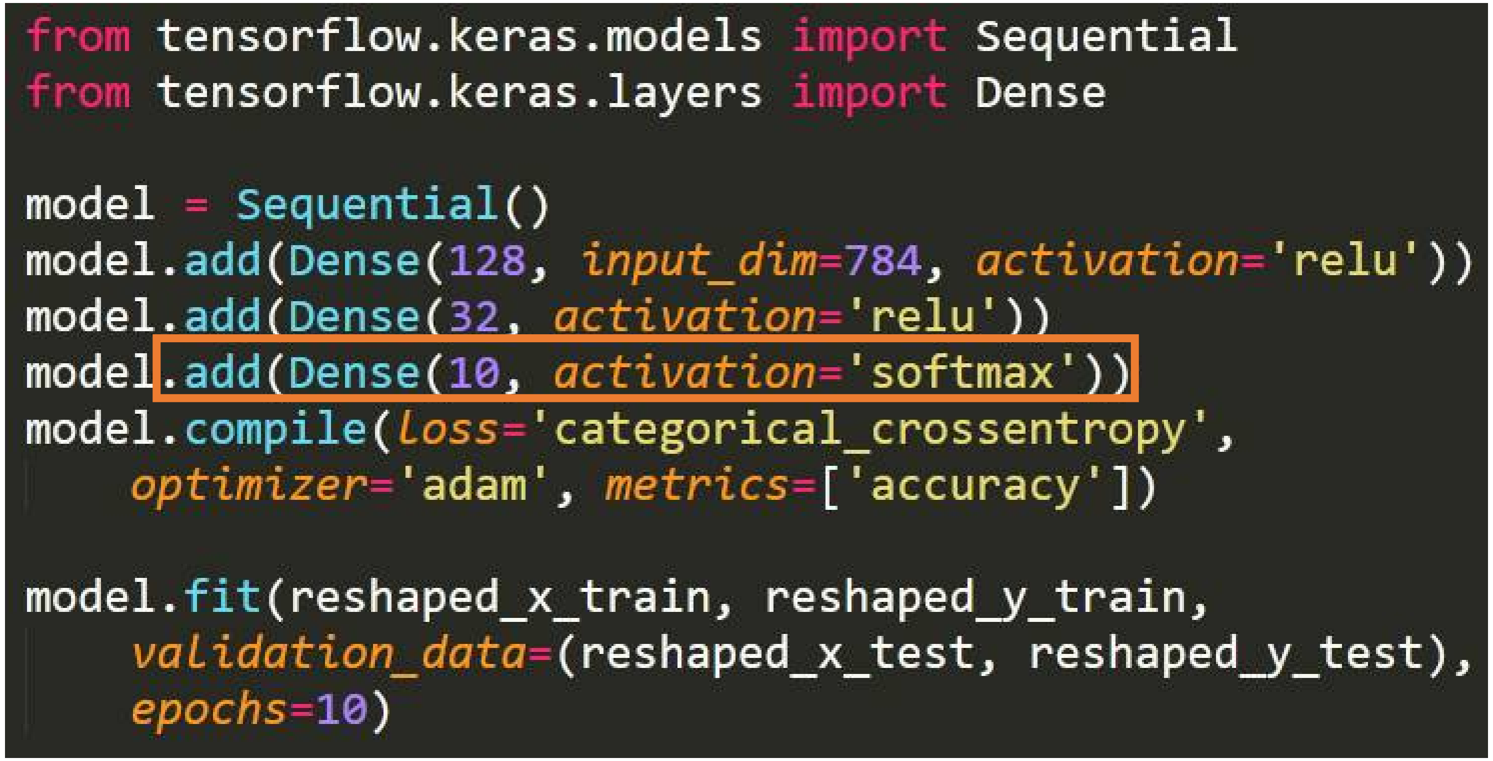
tambahkan hidden layer kedua dengan **32 node** dan activation function ReLU



input layer

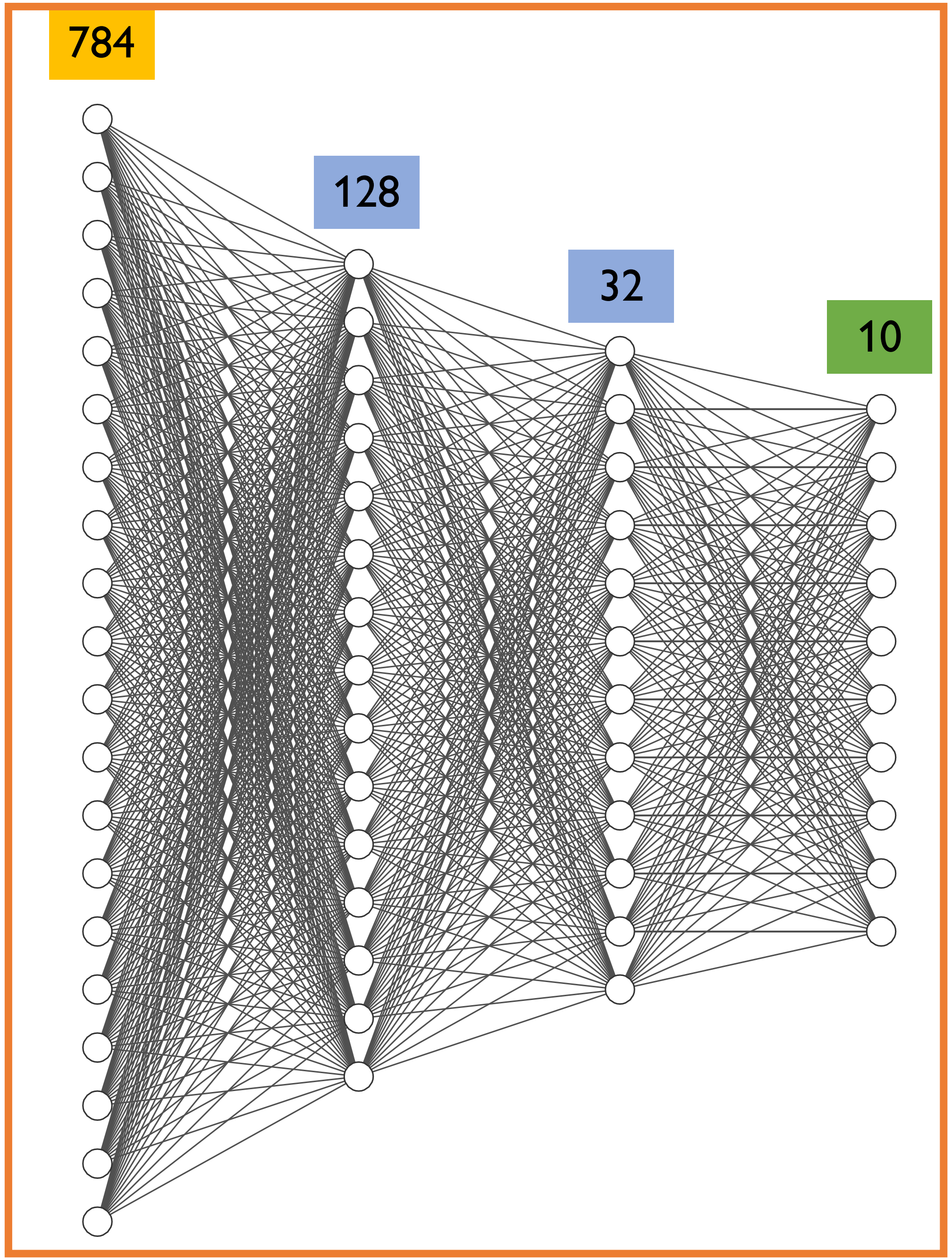
hidden layer

output layer



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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

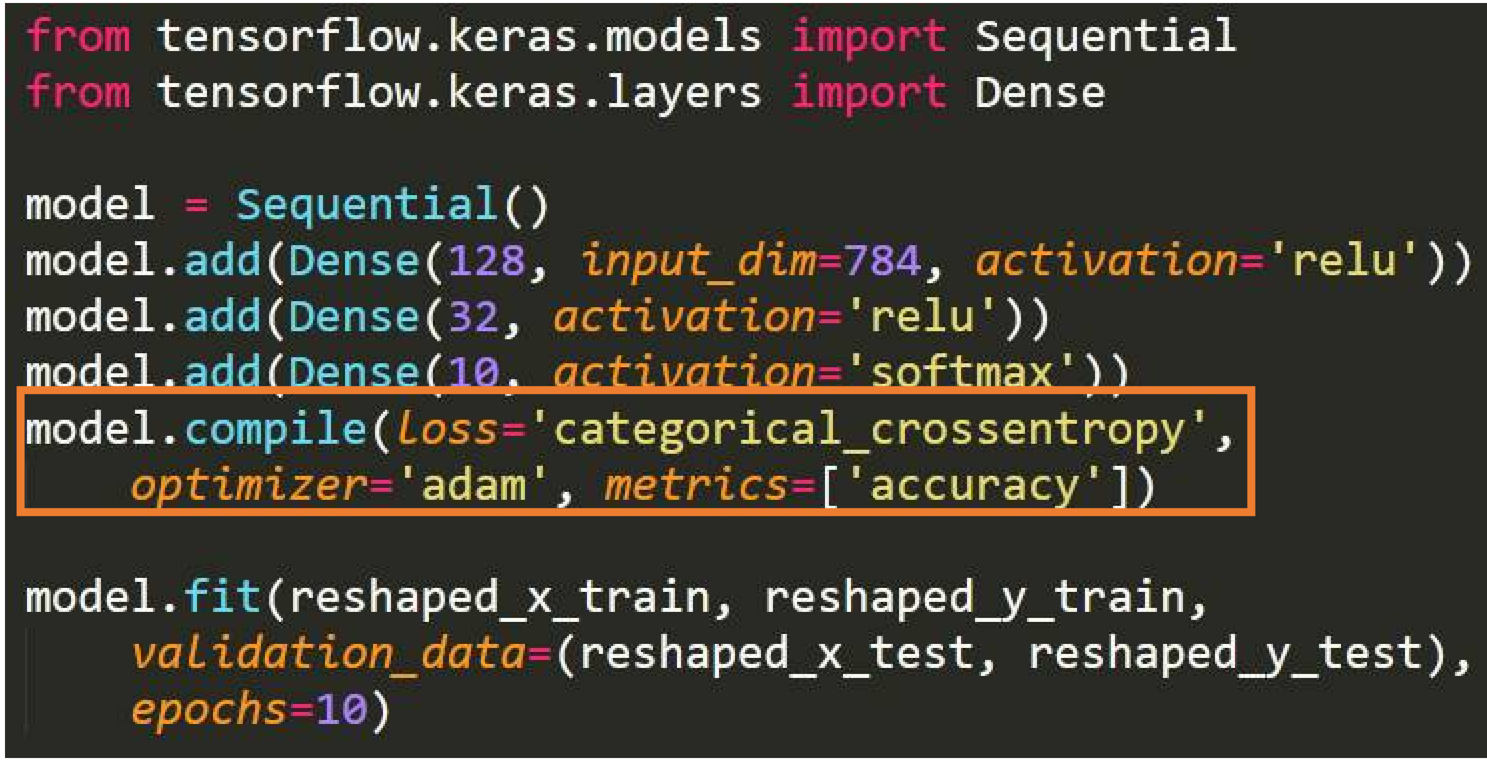
note :
pada layer output jumlah node sesuai dengan jumlah kelas. di sini kita menggunakan fungsi aktivasi **Softmax untuk menghitung probabilitas** kelas tiap data.



input layer

hidden layer

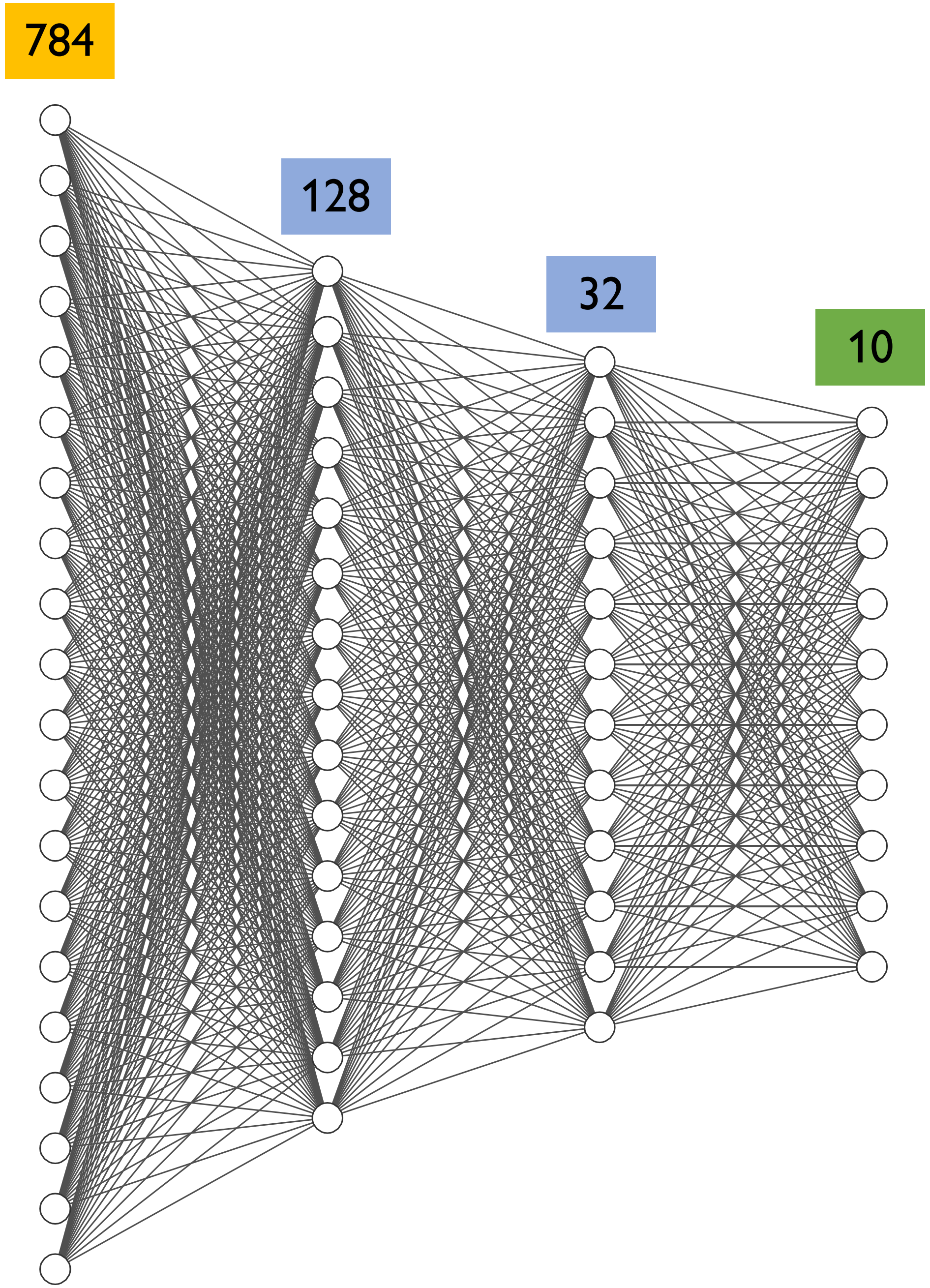
output layer



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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

note : **compile()**

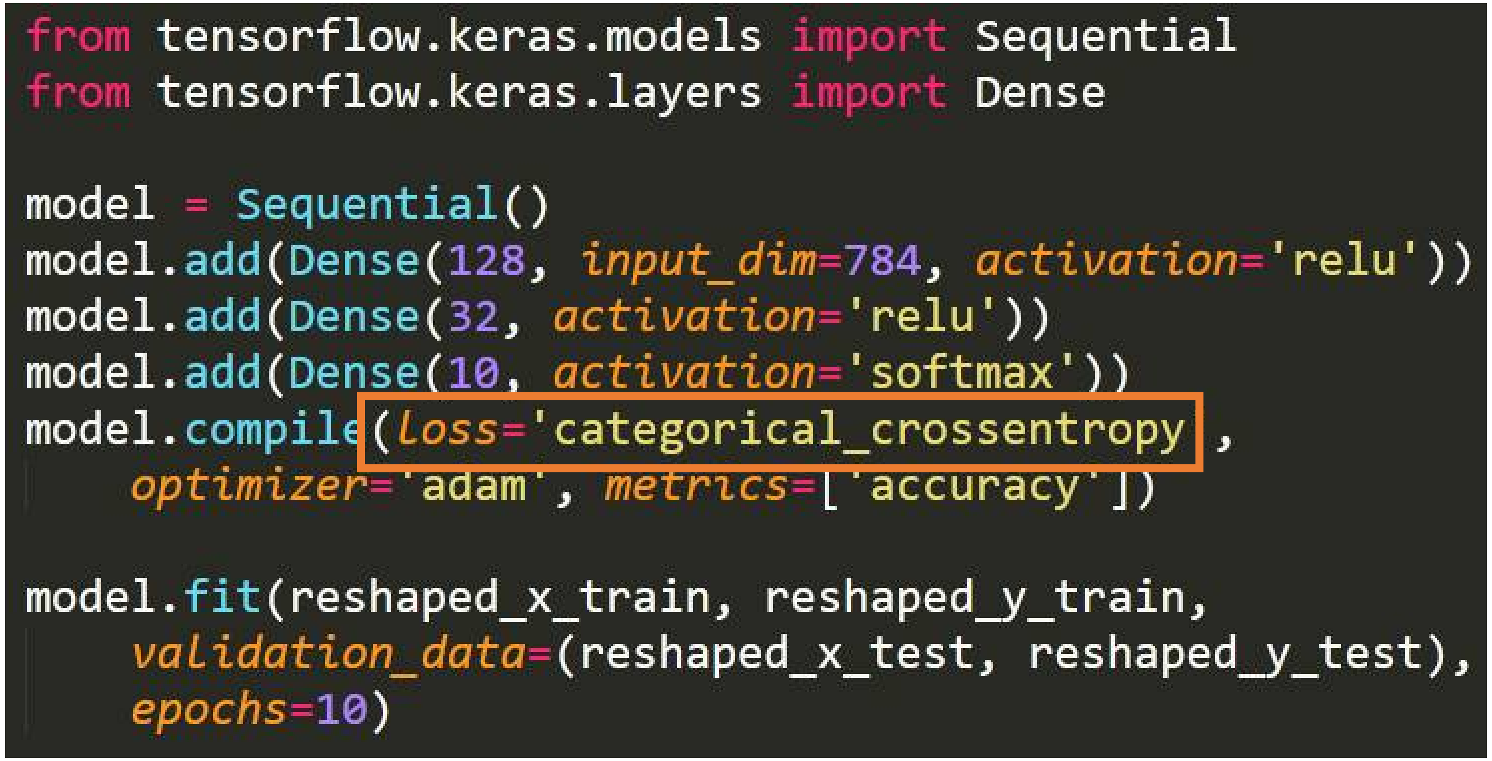
konfigurasi bagaimana model akan ‘belajar’



input layer

hidden layer

output layer



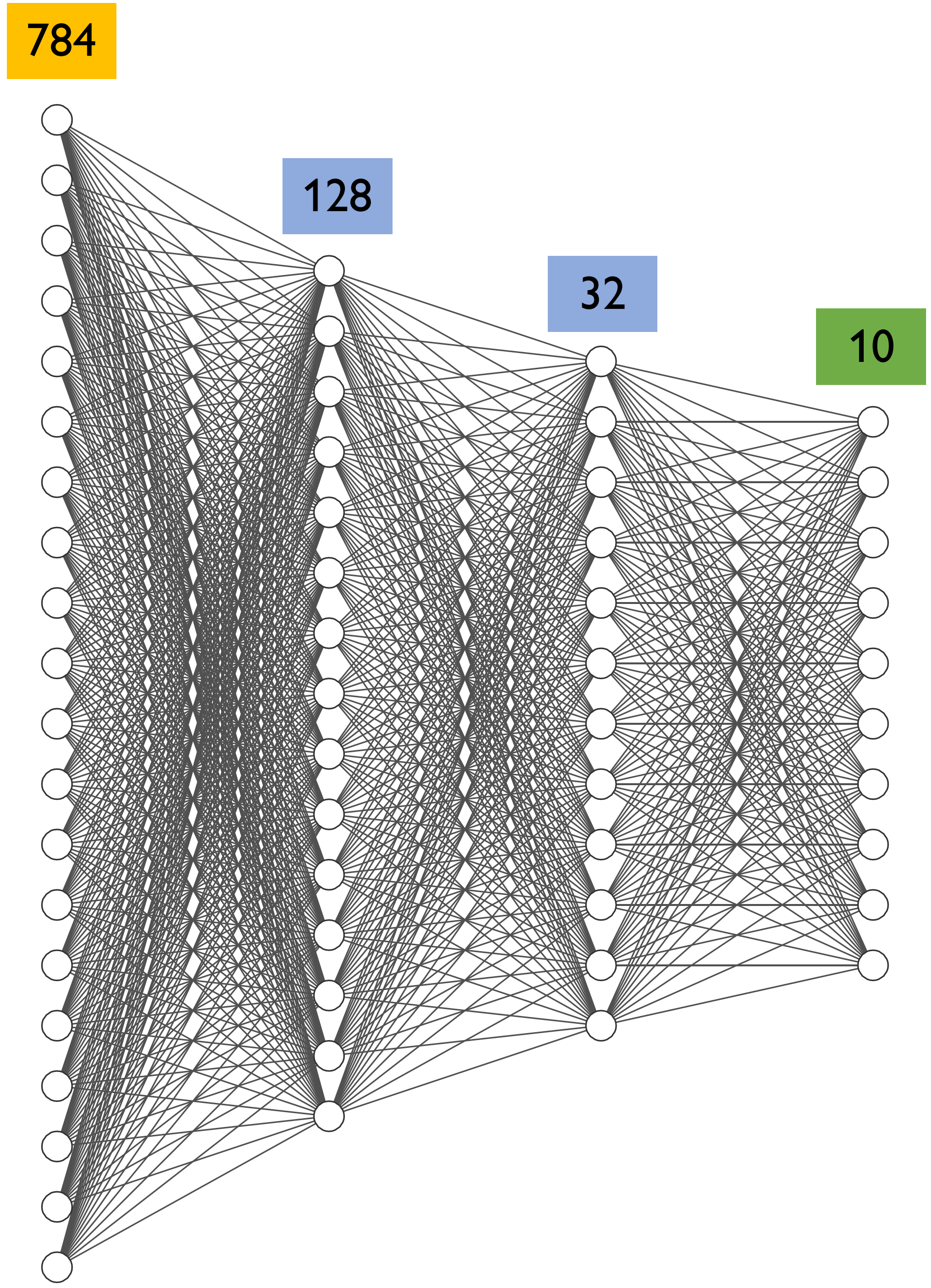
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

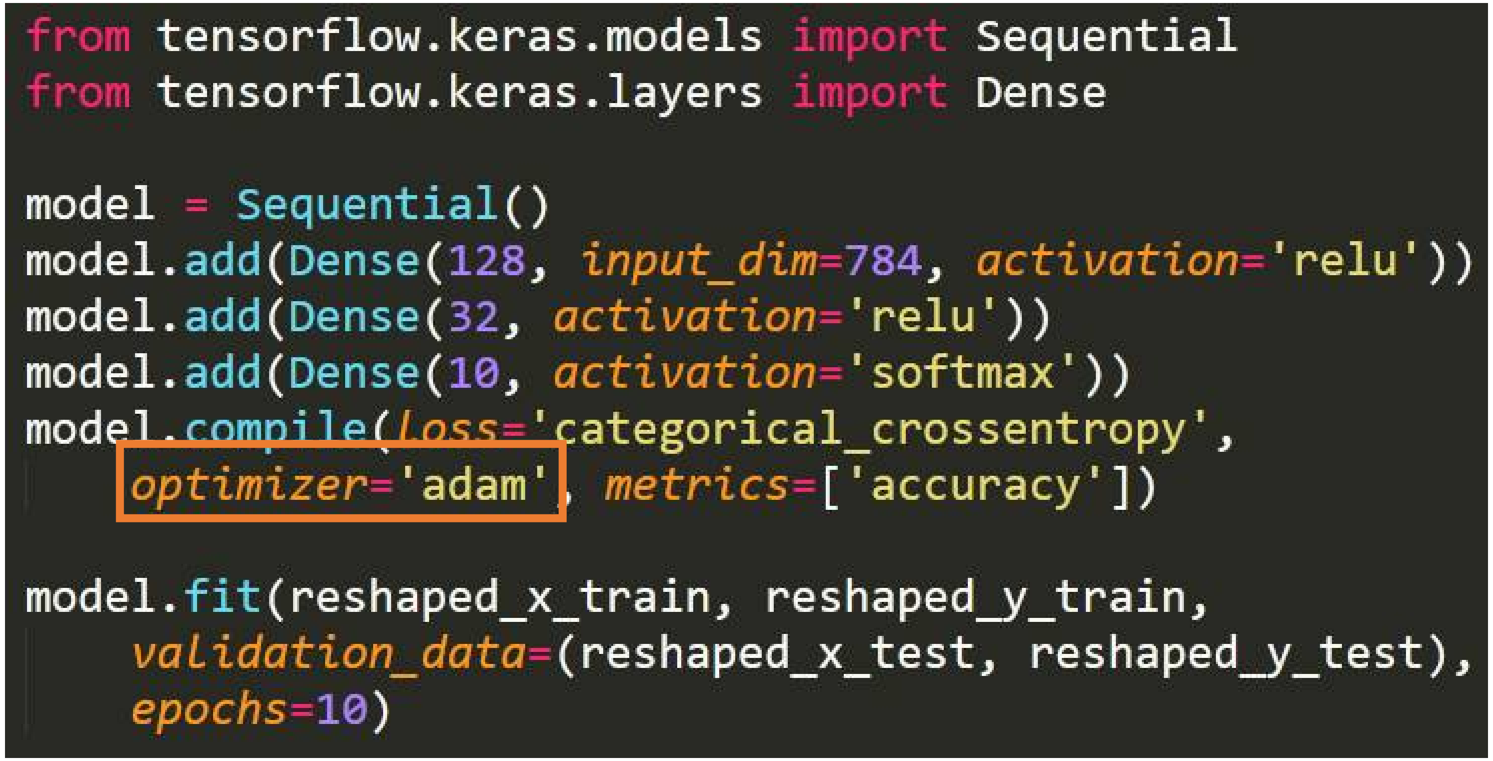
note :
loss adalah fungsi objektif/cara model menghitung perbedaan antara prediksi antara label asli. tujuan training adalah meminiminalkan loss. untuk multi-kelas, digunakan **categorical_crossentropy**

input layer

hidden layer

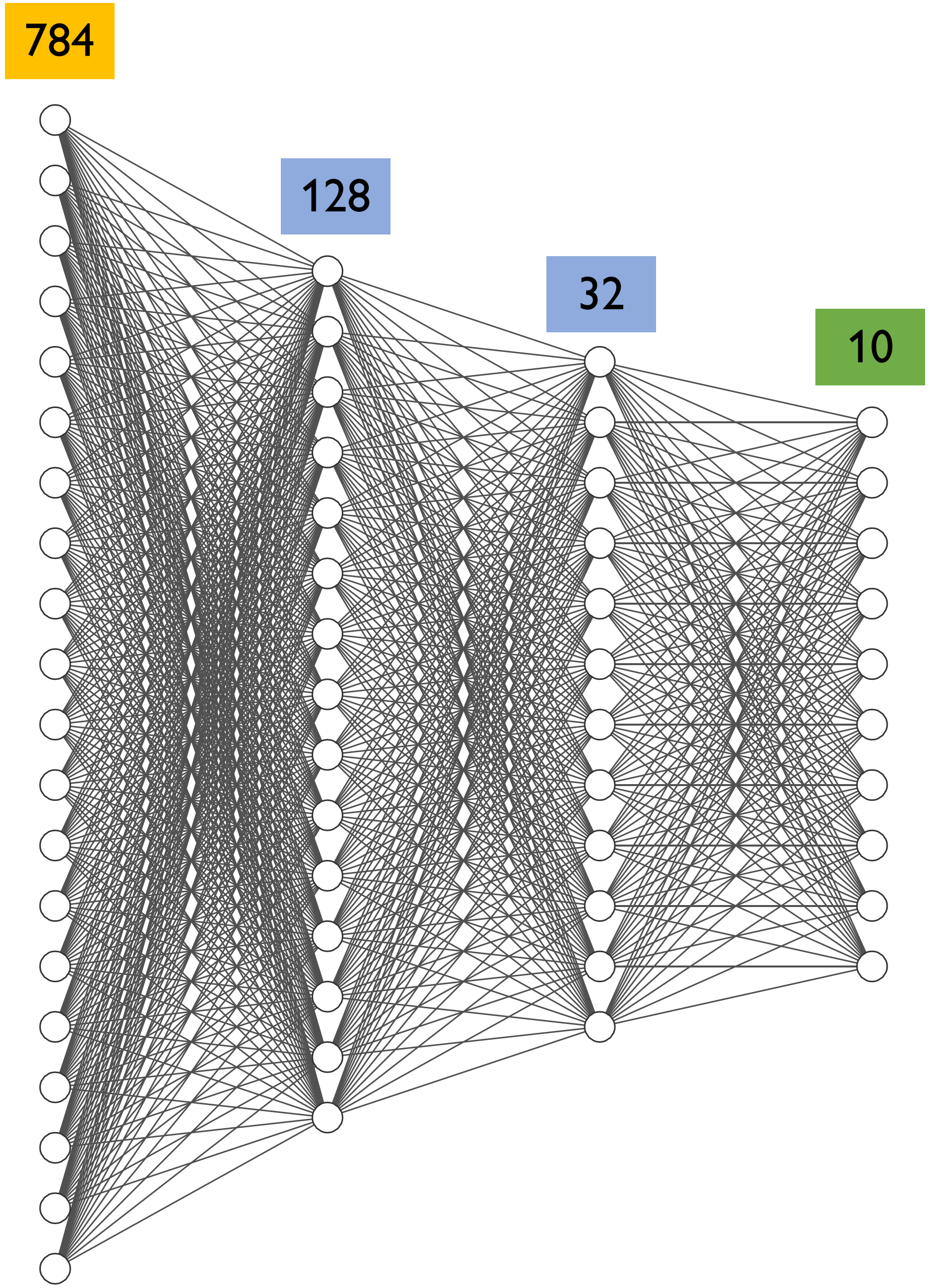
output layer





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

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

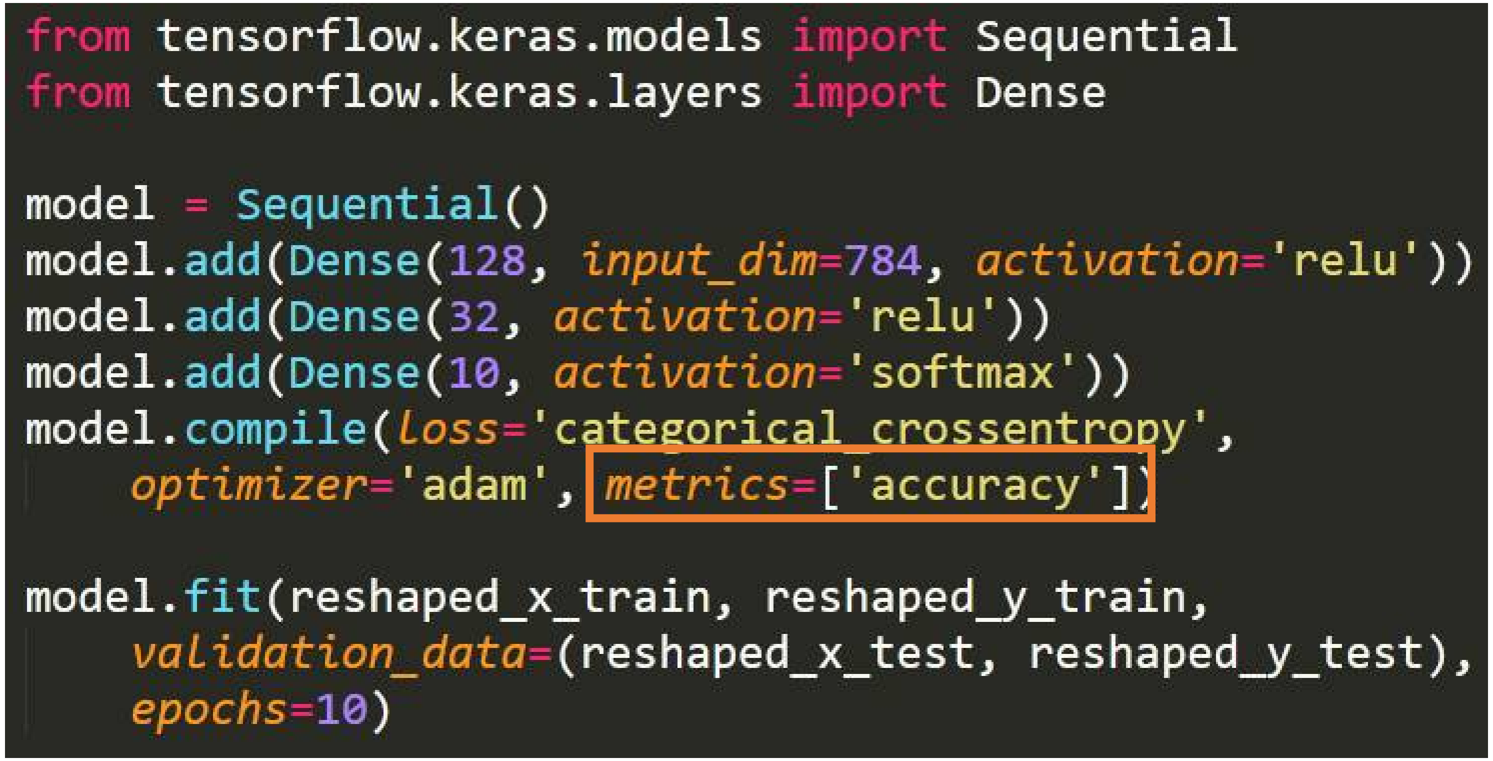
note :
optimizer adalah cara model ANN meminimalisir nilai loss. digunakan metode **Adam**



input layer

hidden layer

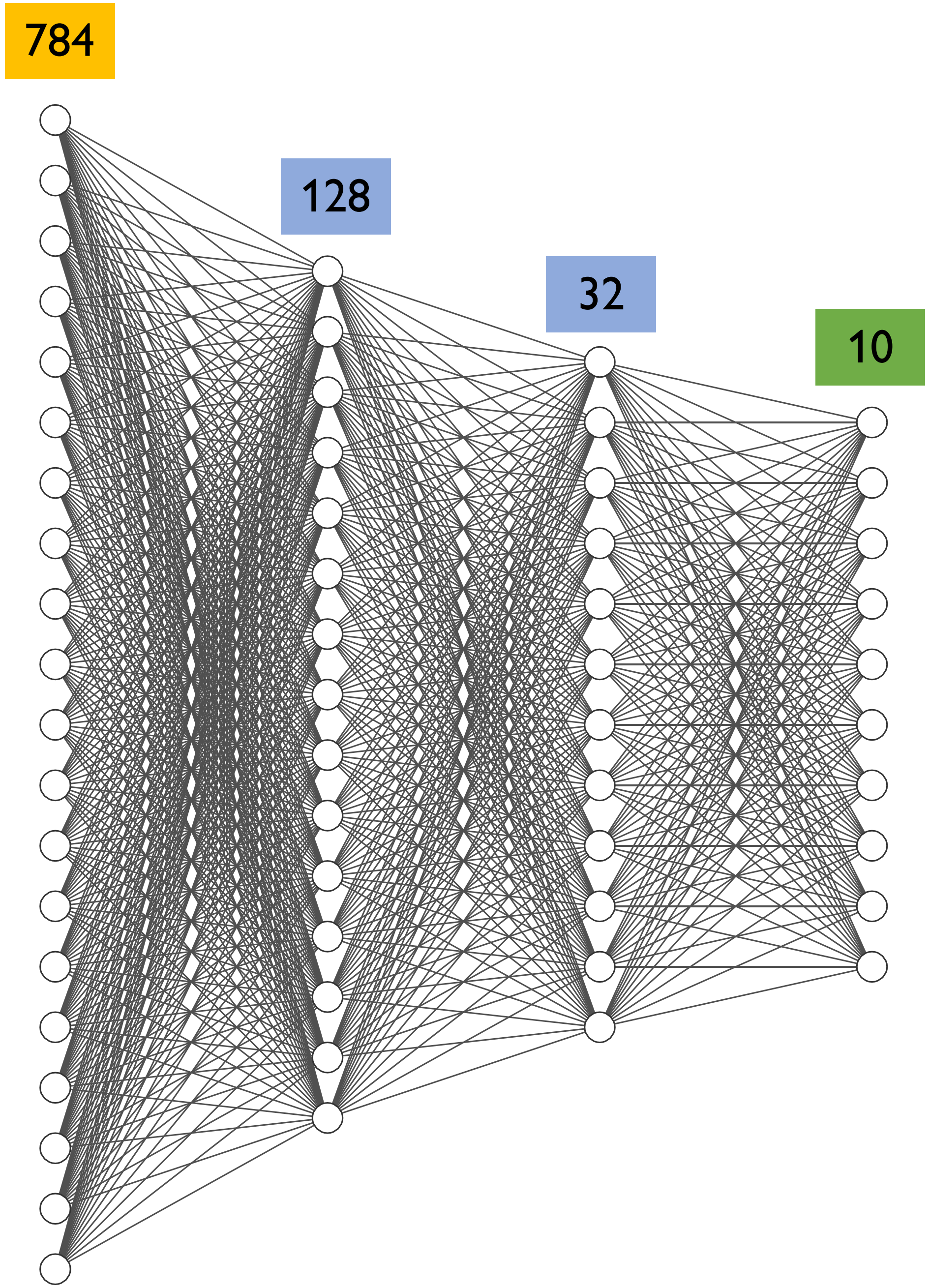
output layer

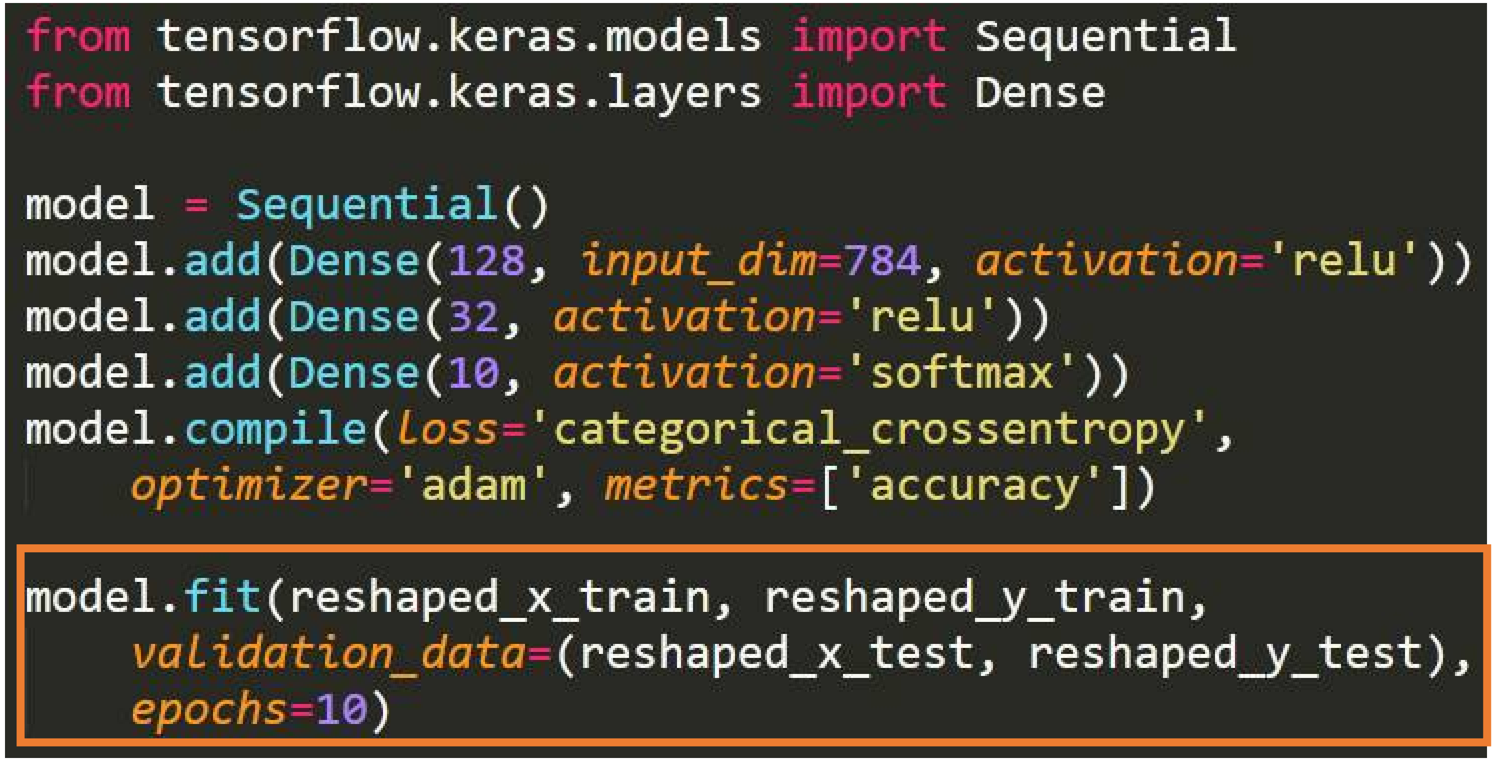


```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

note :

metrics adalah pengukuran evaluasi kinerja model ANN. digunakan **accuracy**

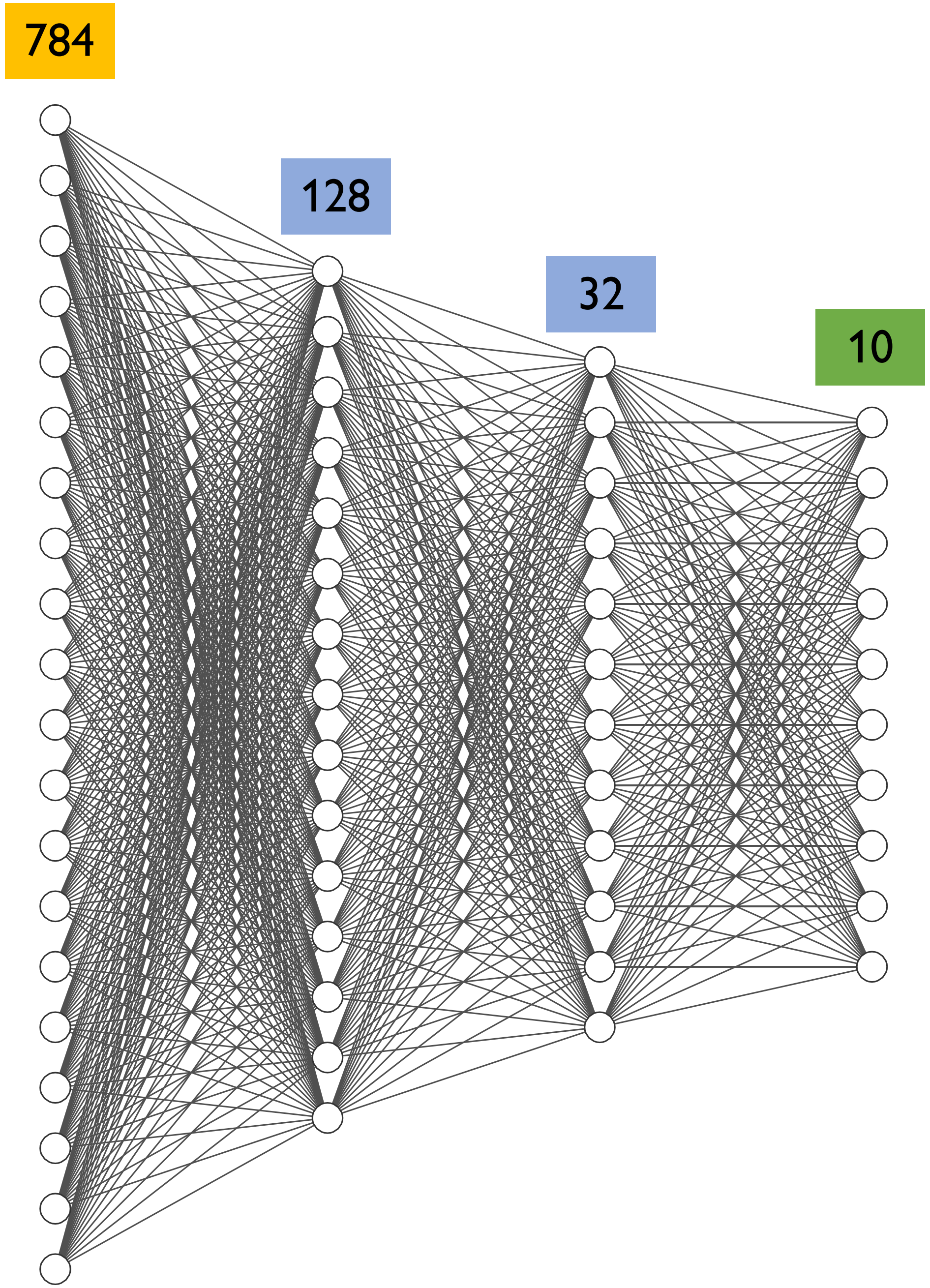




```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

note :

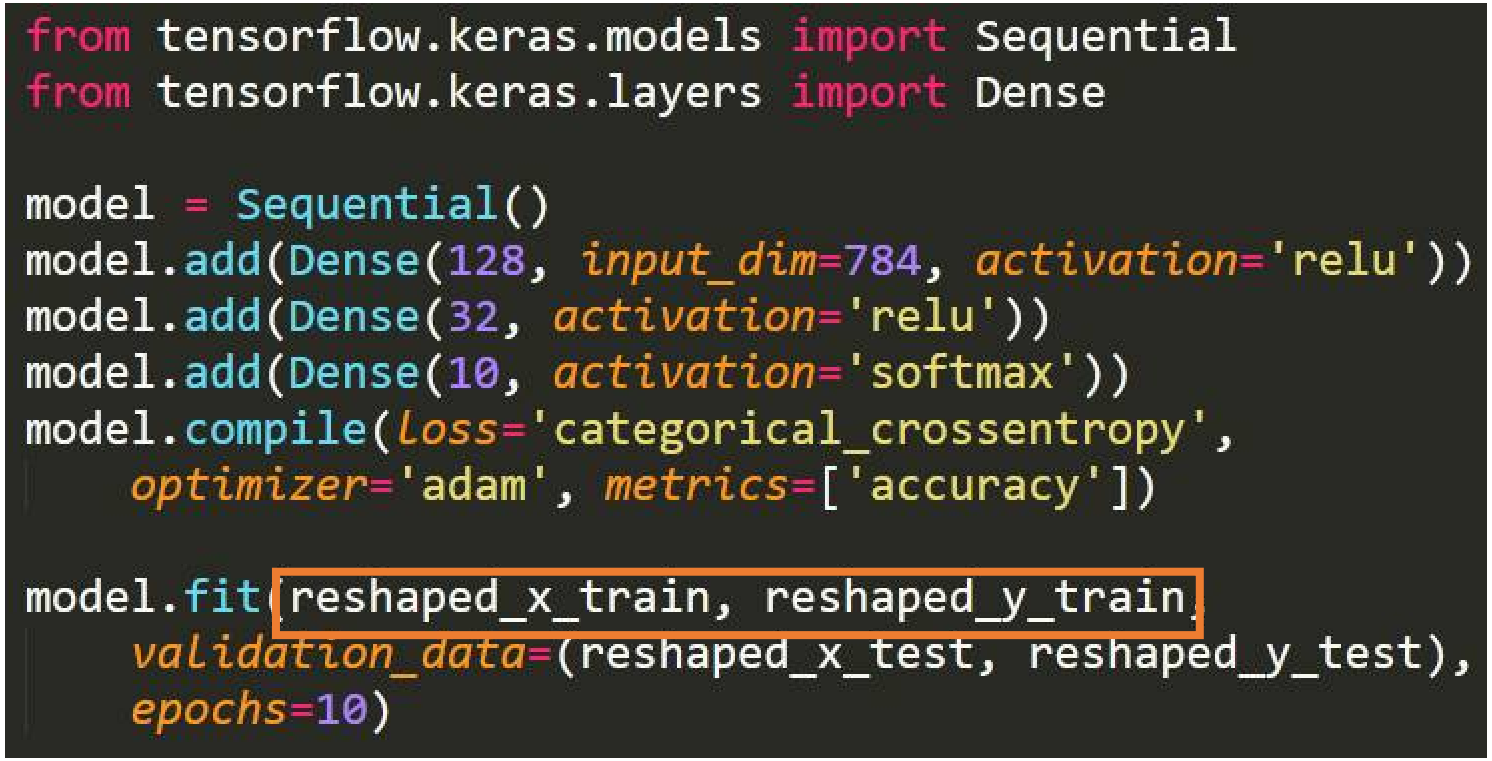
fit merupakan method untuk mentraining model yang kita buat



input layer

hidden layer

output layer



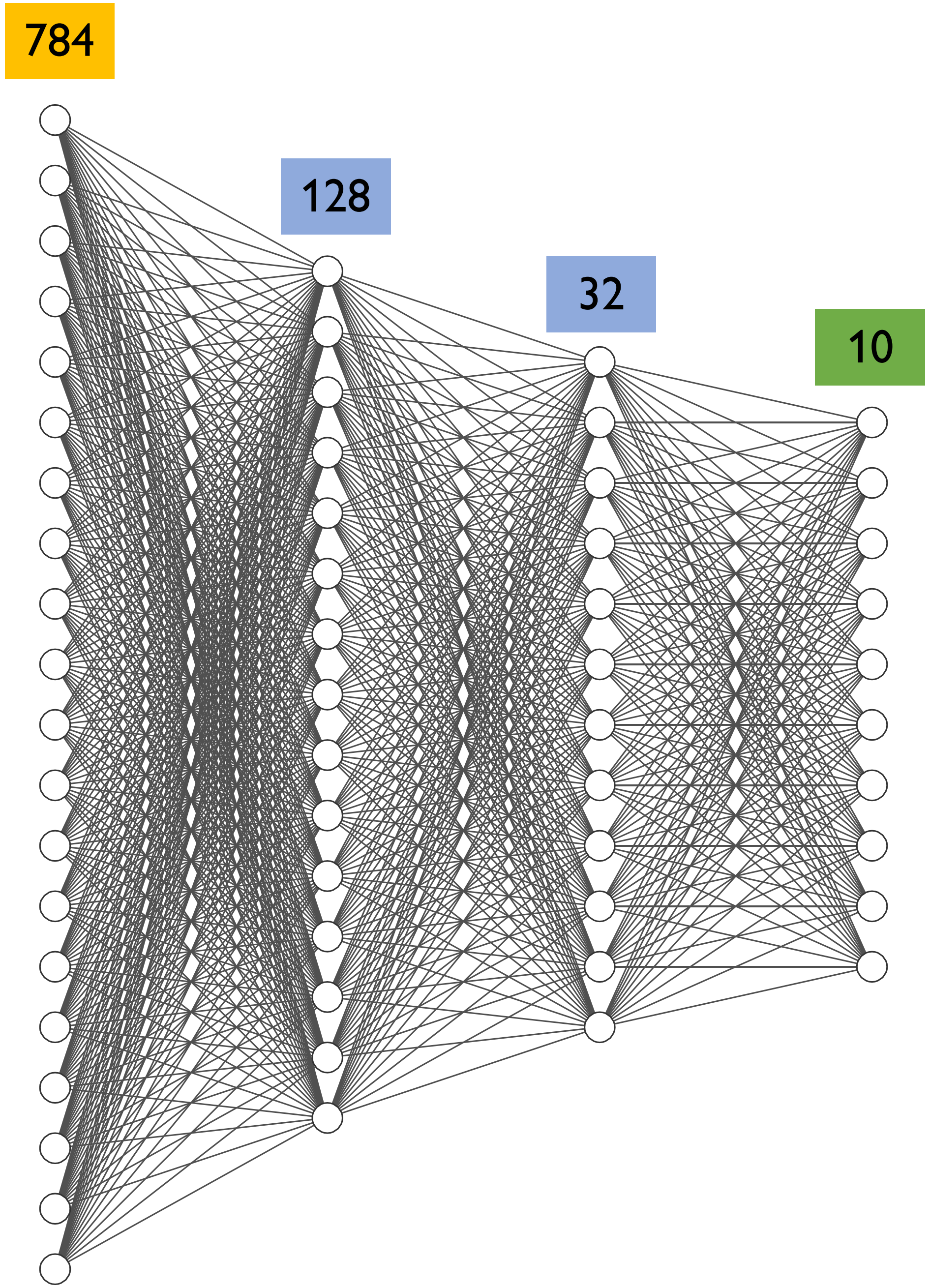
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

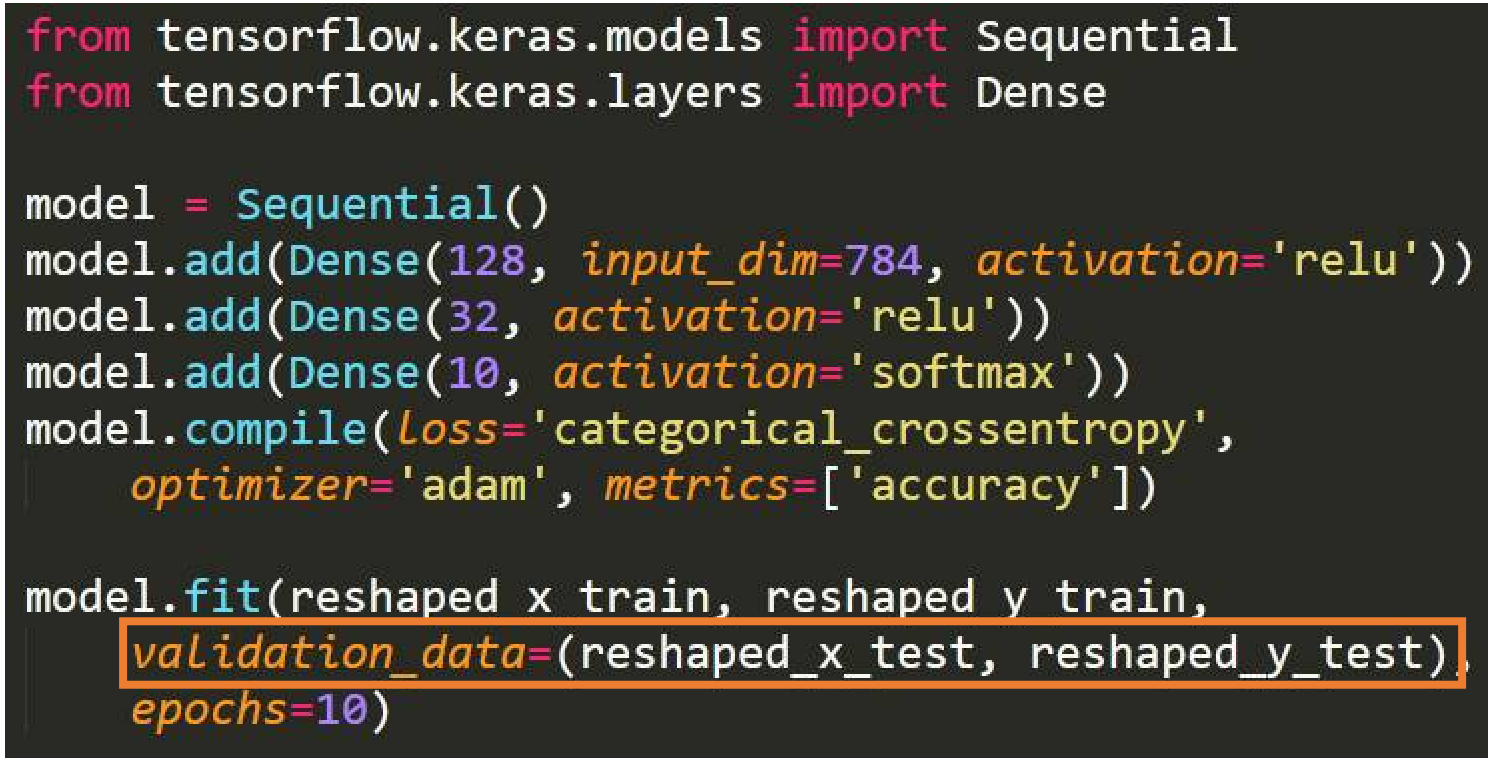
note :
data yang digunakan untuk training adalah *reshaped_x_train* dengan label *reshaped_y_train.* patut diperhatikan ini adalah positional parameter/arguments, artinya data harus dideklarasikan terlebih dahulu

input layer

hidden layer

output layer





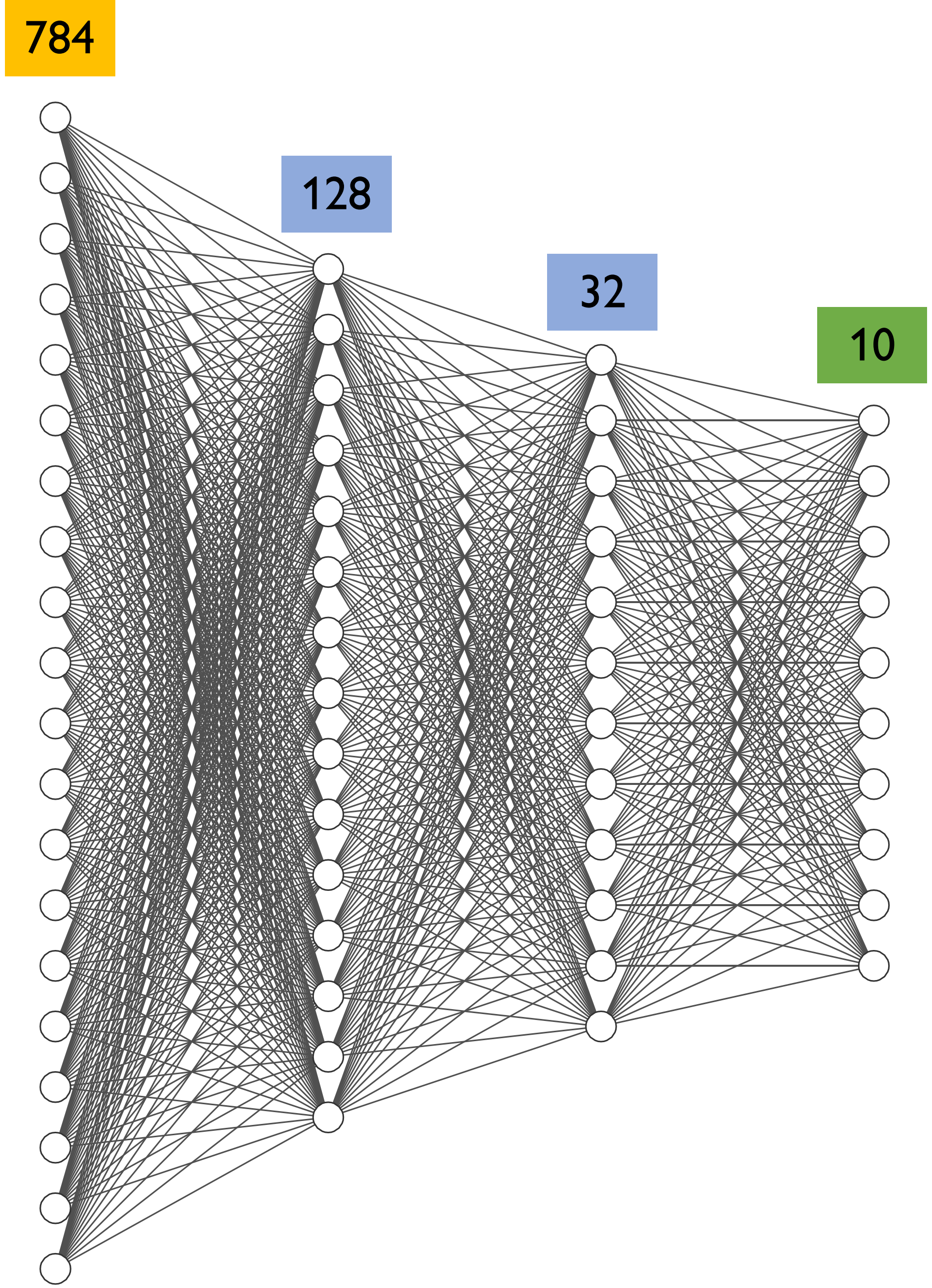
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped x train, reshaped y train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

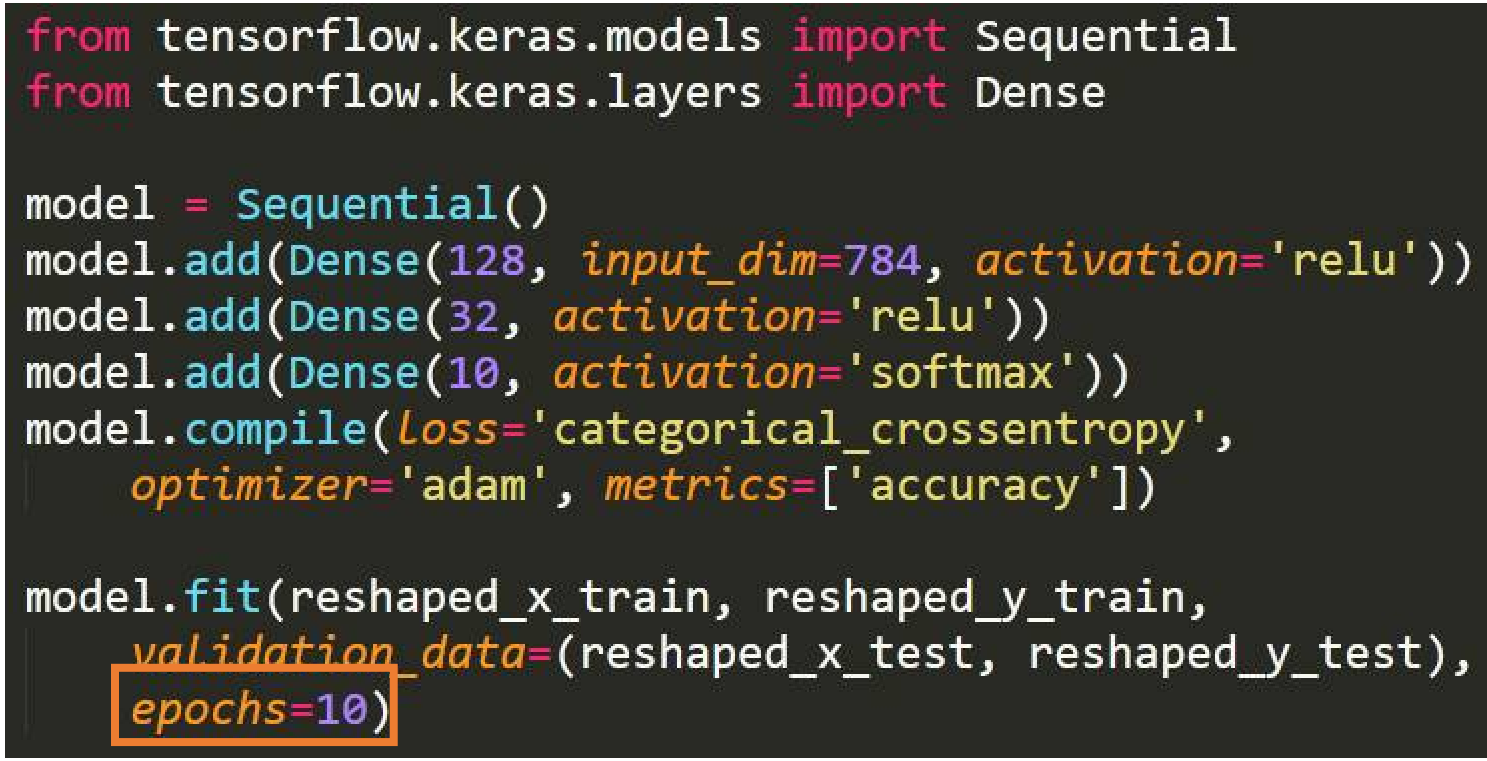
note :
setiap kali model belajar semua data training, model yang terbentuk diuji performanya dengan data validasi, yaitu data *reshaped_x_test* dengan label *reshaped_y_test*

input layer

hidden layer

output layer



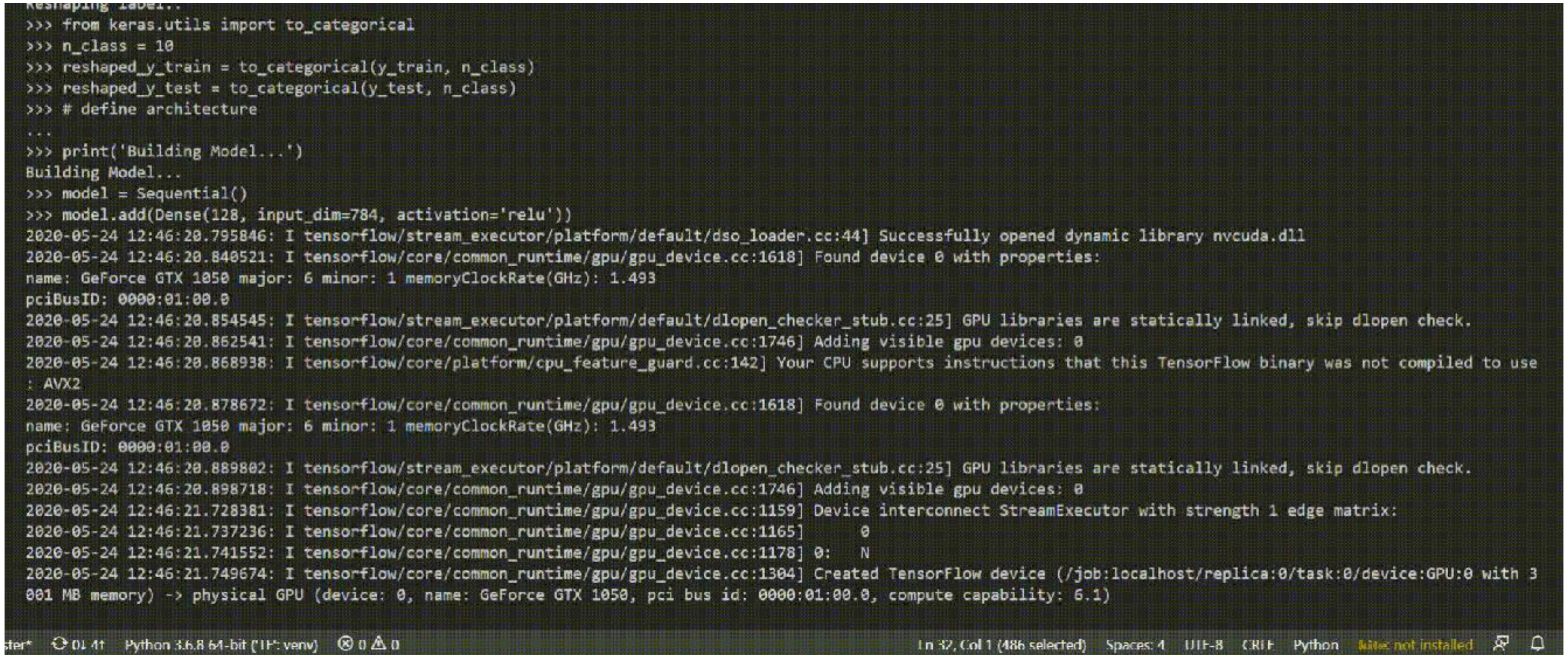


```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_dim=784, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])

model.fit(reshaped_x_train, reshaped_y_train,
    validation_data=(reshaped_x_test, reshaped_y_test),
    epochs=10)
```

note :
satu epoch menandakan model telah belajar semua data training sebanyak satu kali. di sini dituliskan epoch=10, ini artinya model belajar semua data training sebanyak 10 kali

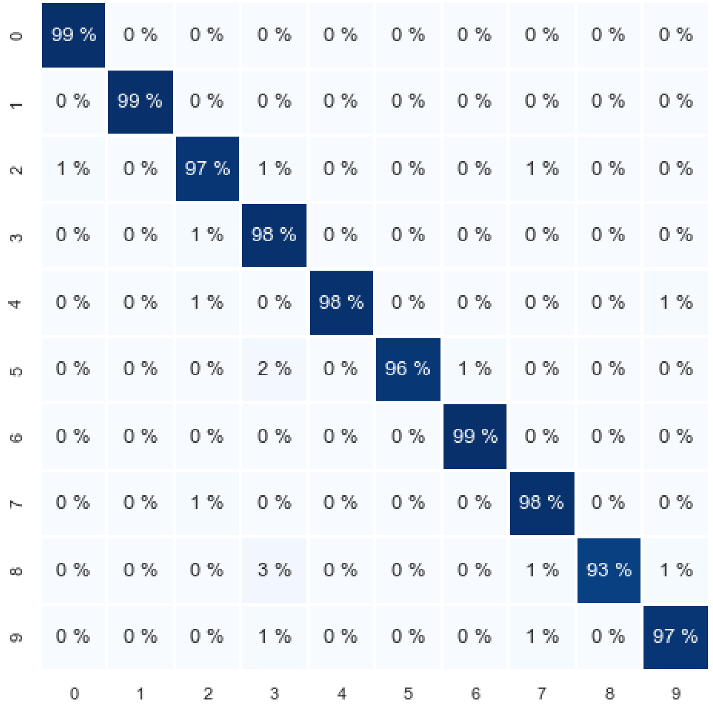
Training, Evaluation & Save Model

training log



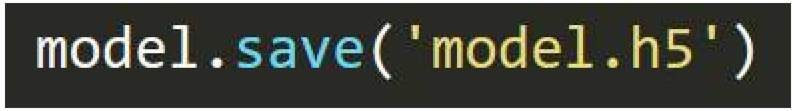
```
Reshaping label..
>>> from keras.utils import to_categorical
>>> n_class = 10
>>> reshaped_y_train = to_categorical(y_train, n_class)
>>> reshaped_y_test = to_categorical(y_test, n_class)
>>> # define architecture
...
>>> print('Building Model...')
Building Model...
>>> model = Sequential()
>>> model.add(Dense(128, input_dim=784, activation='relu'))
2020-05-24 12:46:20.795846: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
2020-05-24 12:46:20.840521: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
name: GeForce GTX 1050 major: 6 minor: 1 memoryClockRate(GHz): 1.493
pciBusID: 0000:01:00.0
2020-05-24 12:46:20.854545: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries are statically linked, skip dlopen check.
2020-05-24 12:46:20.862541: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1746] Adding visible gpu devices: 0
2020-05-24 12:46:20.868938: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use
: AVX2
2020-05-24 12:46:20.878672: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
name: GeForce GTX 1050 major: 6 minor: 1 memoryClockRate(GHz): 1.493
pciBusID: 0000:01:00.0
2020-05-24 12:46:20.889802: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries are statically linked, skip dlopen check.
2020-05-24 12:46:20.898718: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1746] Adding visible gpu devices: 0
2020-05-24 12:46:21.728381: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1159] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-05-24 12:46:21.737236: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1165]      0
2020-05-24 12:46:21.741552: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 0:   N
2020-05-24 12:46:21.749674: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 3
001 MB memory) -> physical GPU (device: 0, name: GeForce GTX 1050, pci bus id: 0000:01:00.0, compute capability: 6.1)
```

confusion matrix



0	99 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
1	0 %	99 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
2	1 %	0 %	97 %	1 %	0 %	0 %	0 %	1 %	0 %	0 %
3	0 %	0 %	1 %	98 %	0 %	0 %	0 %	0 %	0 %	0 %
4	0 %	0 %	1 %	0 %	98 %	0 %	0 %	0 %	0 %	1 %
5	0 %	0 %	0 %	2 %	0 %	96 %	1 %	0 %	0 %	0 %
6	0 %	0 %	0 %	0 %	0 %	0 %	99 %	0 %	0 %	0 %
7	0 %	0 %	1 %	0 %	0 %	0 %	0 %	98 %	0 %	0 %
8	0 %	0 %	0 %	3 %	0 %	0 %	0 %	1 %	93 %	1 %
9	0 %	0 %	0 %	1 %	0 %	0 %	0 %	1 %	0 %	97 %
	0	1	2	3	4	5	6	7	8	9

saving model



```
model.save('model.h5')
```

pertanyaan/troubleshooting silahkan buat di channel Diskusi Teams