MACHINE LEARNING PARADIGM

- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
  - Clustering
  - Anomaly detection
- Reinforcement Learning

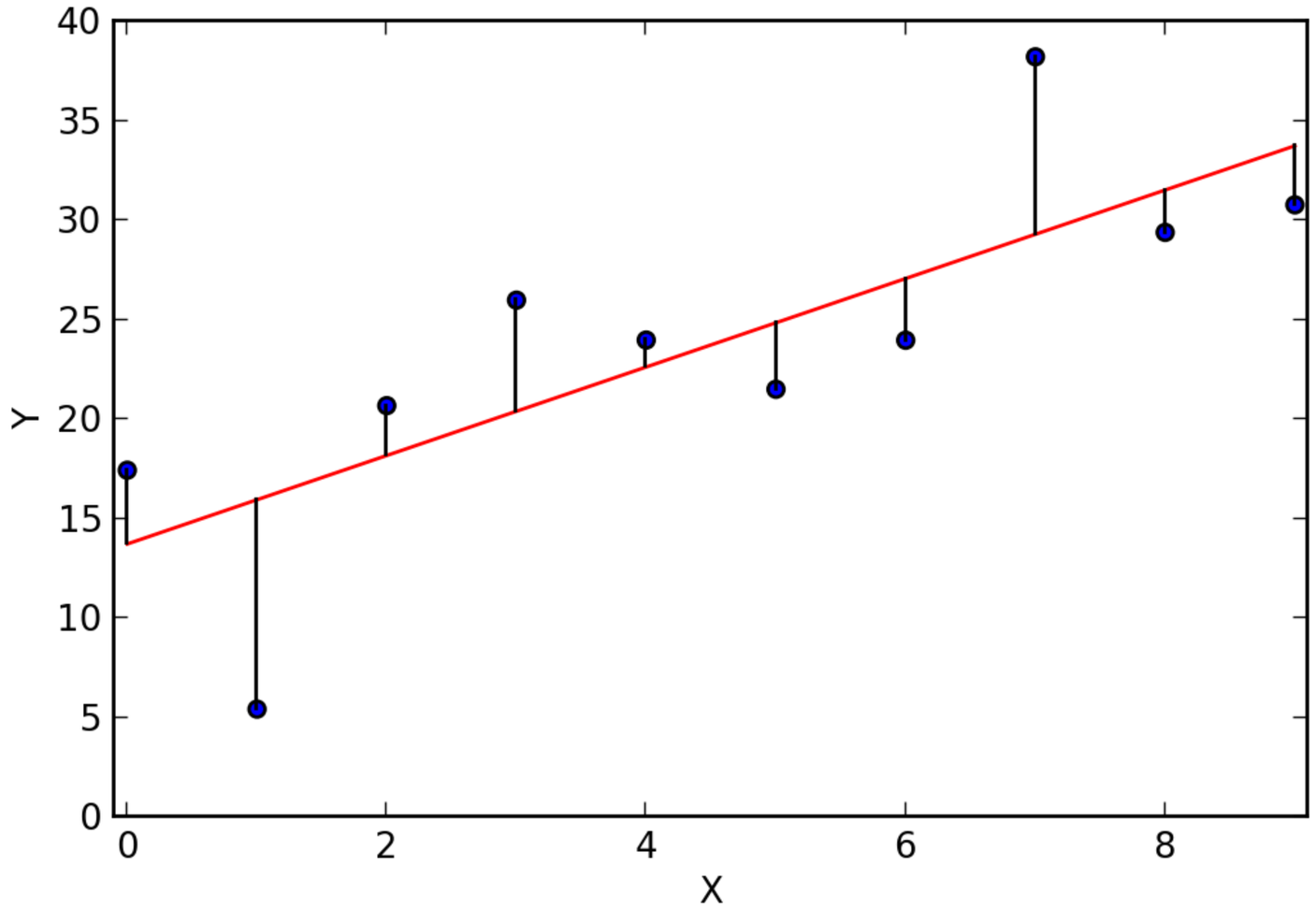
MACHINE LEARNING APPROACH

- Support Vector Machine
- Artificial Neural Network
  - Deep learning
- Decision Tree
- ...

SUPERVISED LEARNING

- Given a set of N training examples of the form $\{(x_1, y_1), \ldots, (x_N,\ y_N)\}\{(x_1, y_1), \ldots, (x_N,\ y_N)\}$
- Define a scoring function $f : X \times Y \to \mathbb{R}$
- Seek a function $g : X \to Y$, which returning the y value that gives the highest score: $g(x) = \arg\max_y f(x, y)$.

LINEAR REGRESSION



LINEAR REGRESSION

$$g = \beta_0 + \beta_1 \times X$$

$$f = \sum_{i=1}^{N} (g(x_i) - Y_i)^2$$

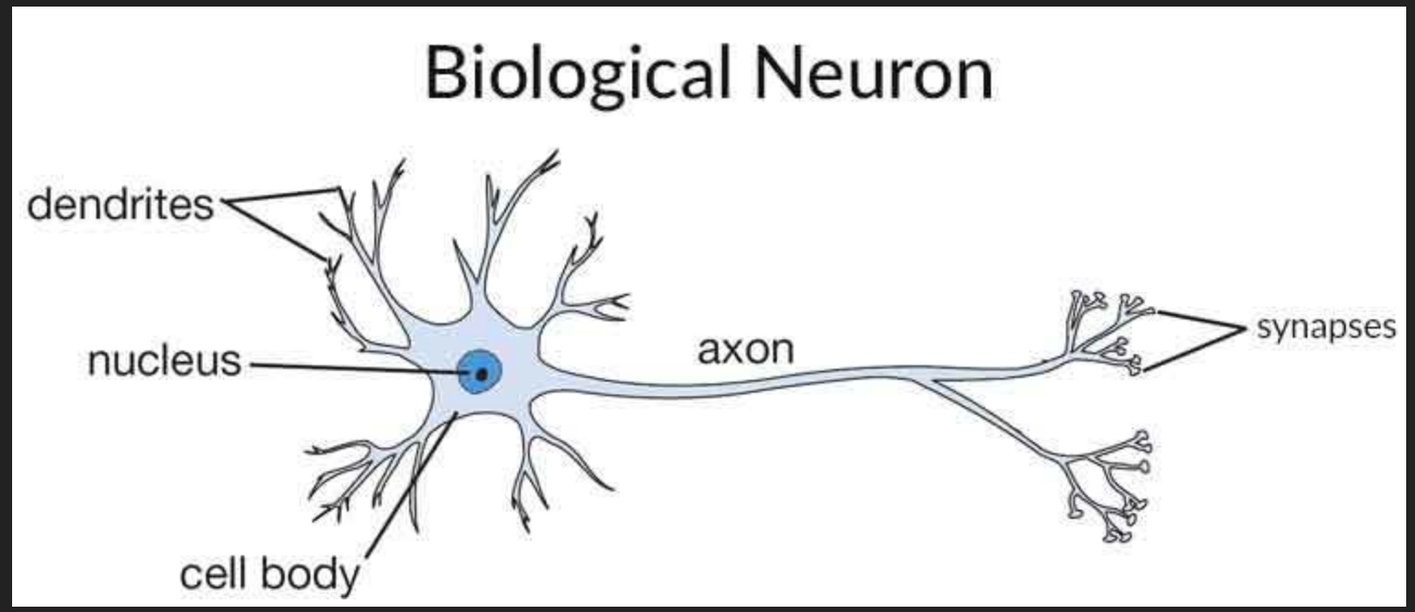
Seek β_0 and β_1, which minimize f

$$f = \sum (\beta_0 + \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_n \end{bmatrix} \times \beta_1 - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \\ y_n \end{bmatrix})^2$$

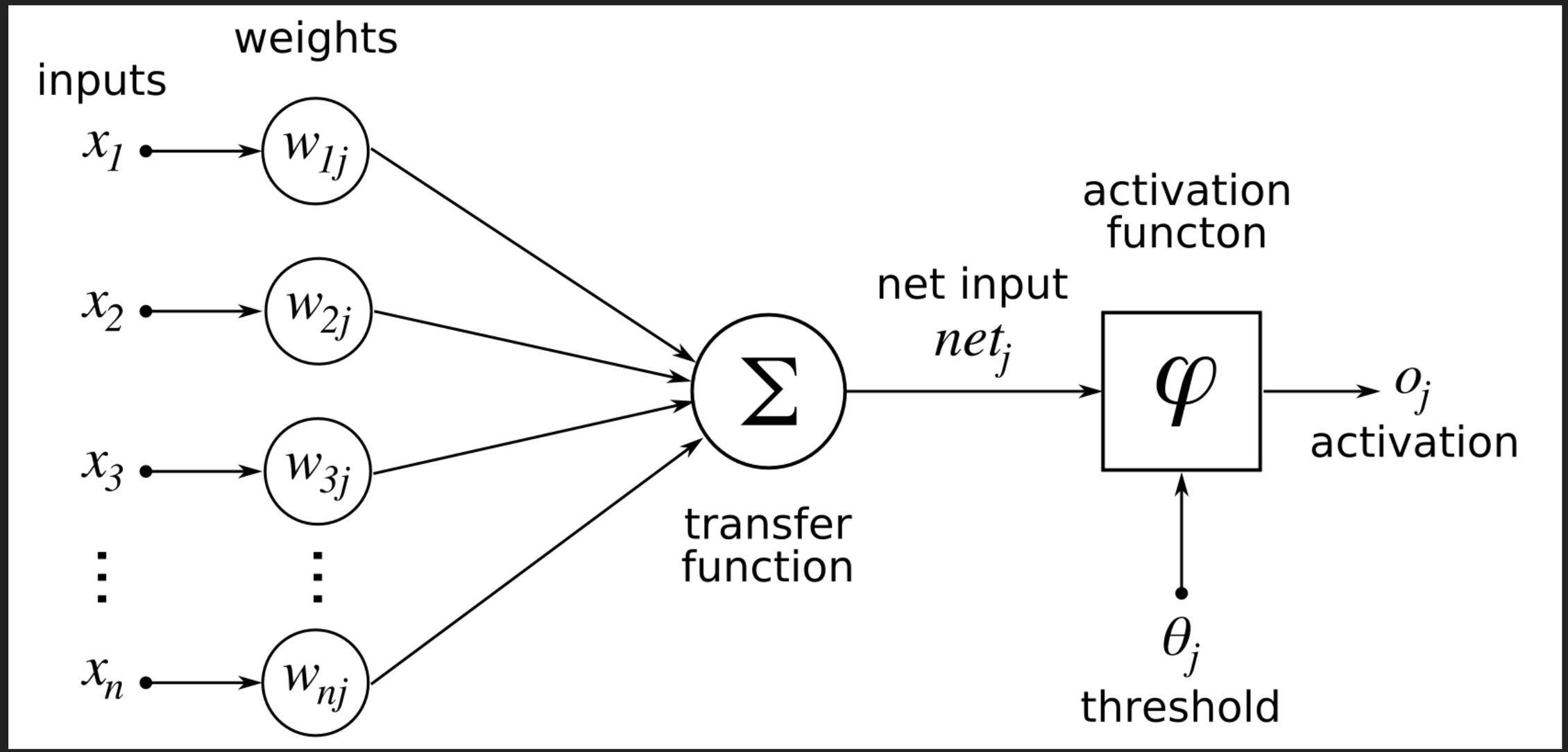
$$= ?$$

ARTIFICIAL NEURAL NETWORK

BIOLOGICAL NEURON

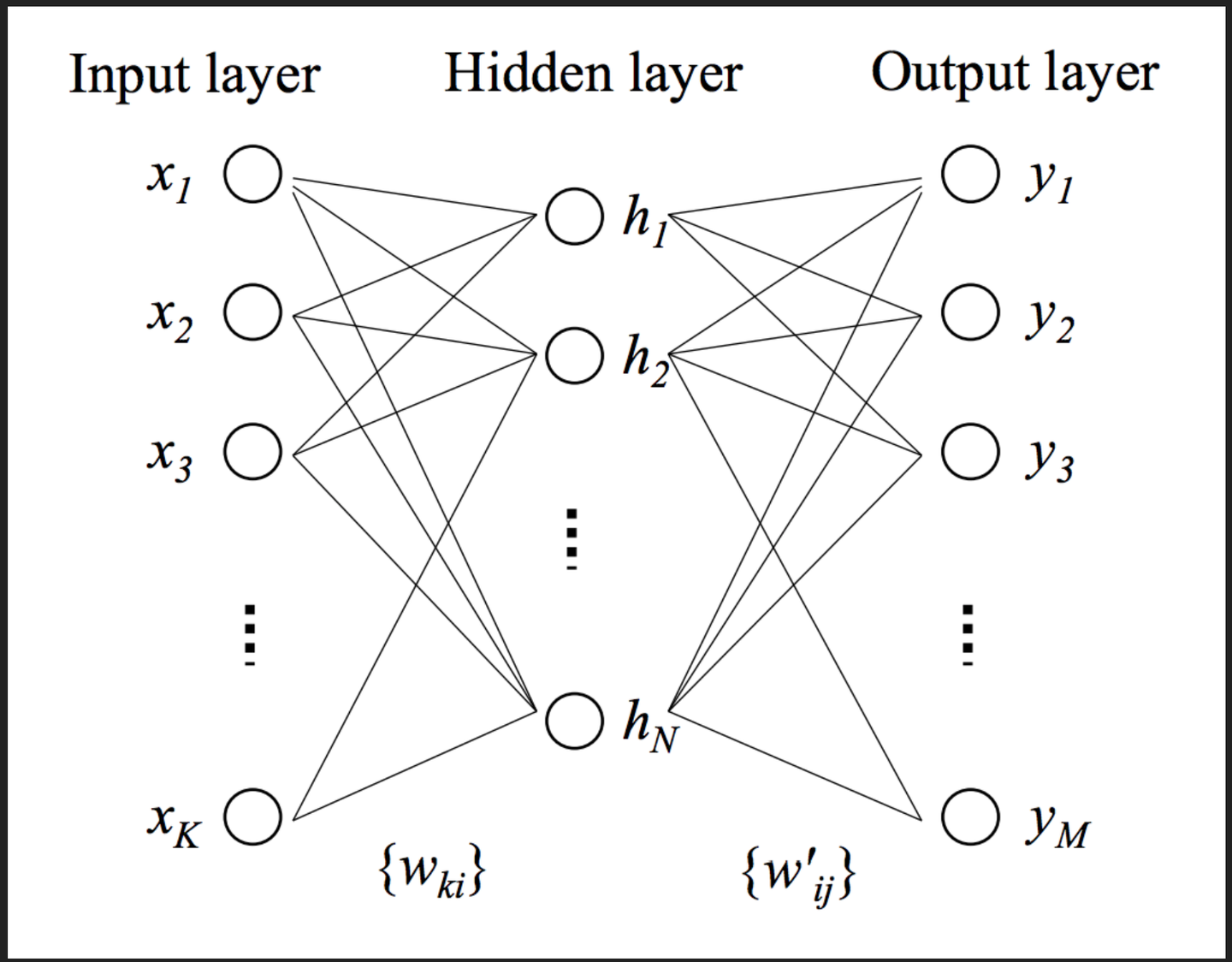


MATHEMATICAL NEURON

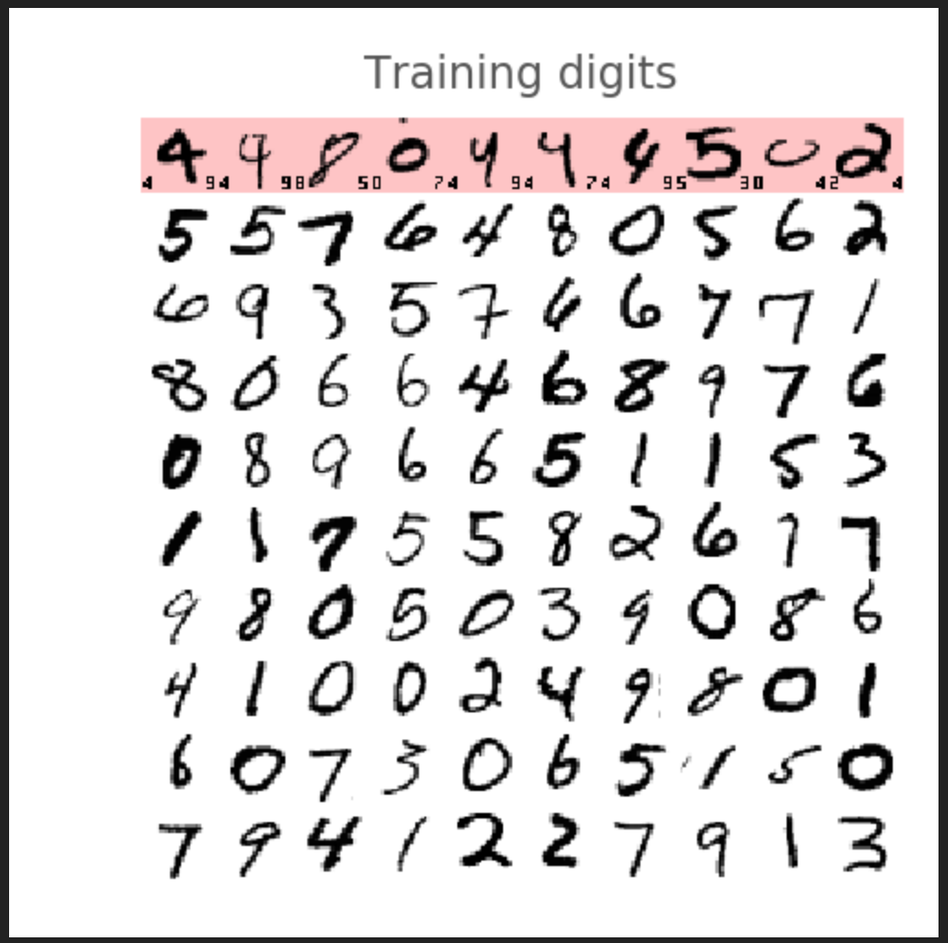


$$f = \psi(\sum(X \times W + b))$$

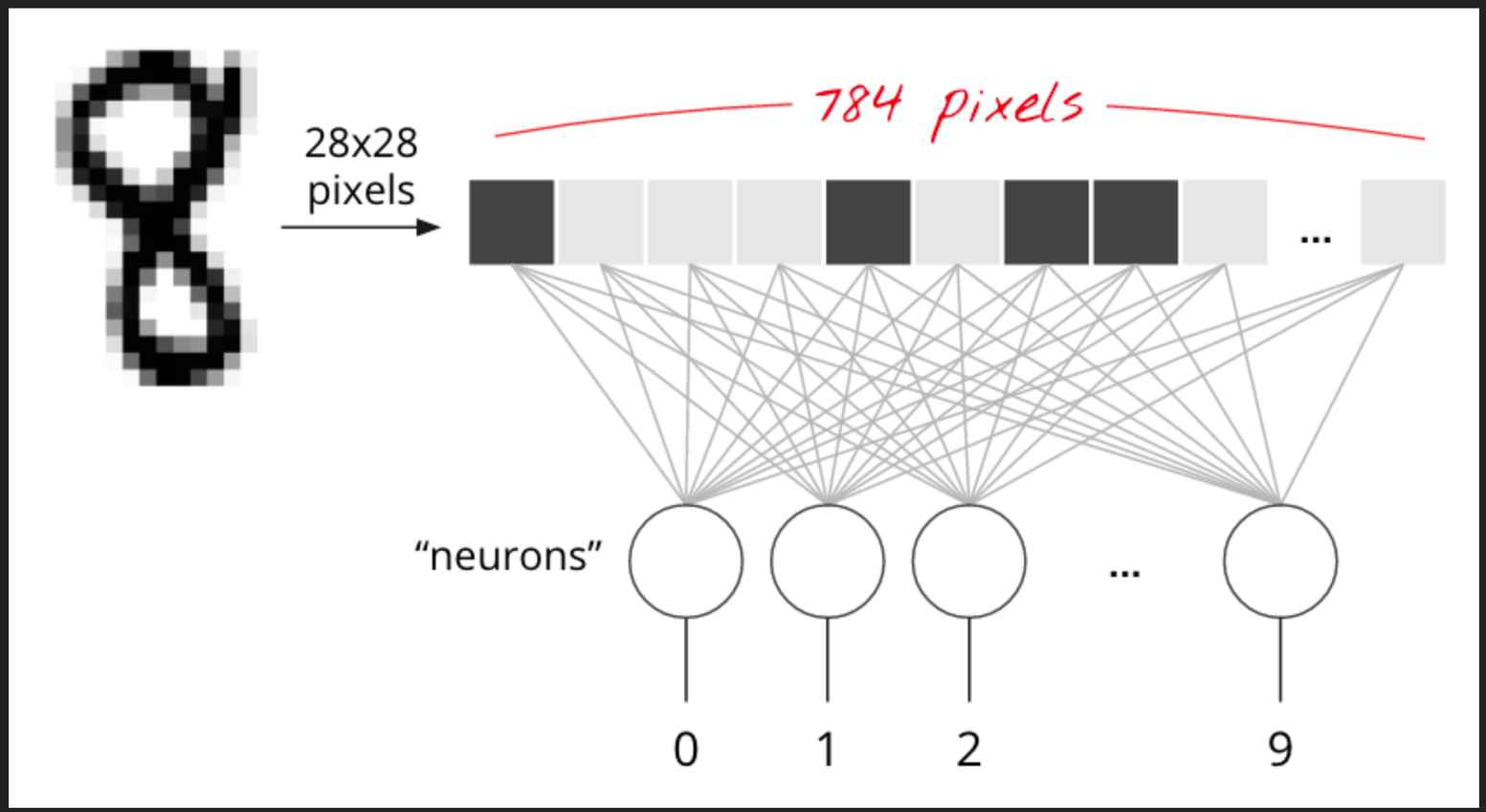
NEURAL NETWORK



HANDWRITTEN DIGITS CLASSIFICATION



1-LAYER NEURAL NETWORK



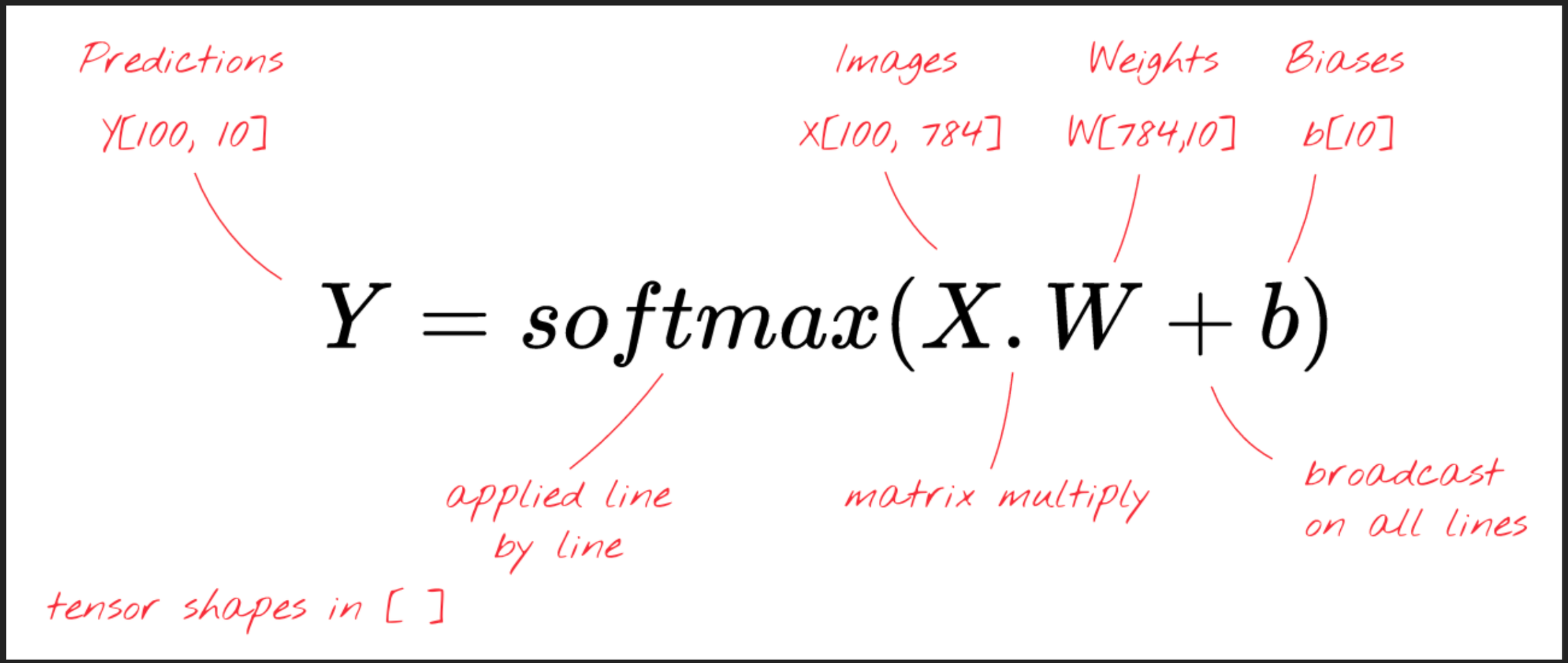
1-LAYER NEURAL NETWORK - WEIGHT

$$W = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \dots & w_{0,9} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \dots & w_{1,9} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \dots & w_{2,9} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \dots & w_{3,9} \\ \dots & & & & & \\ w_{783,0} & w_{783,1} & w_{783,2} & w_{783,3} & \dots & w_{783,9} \end{bmatrix}$$

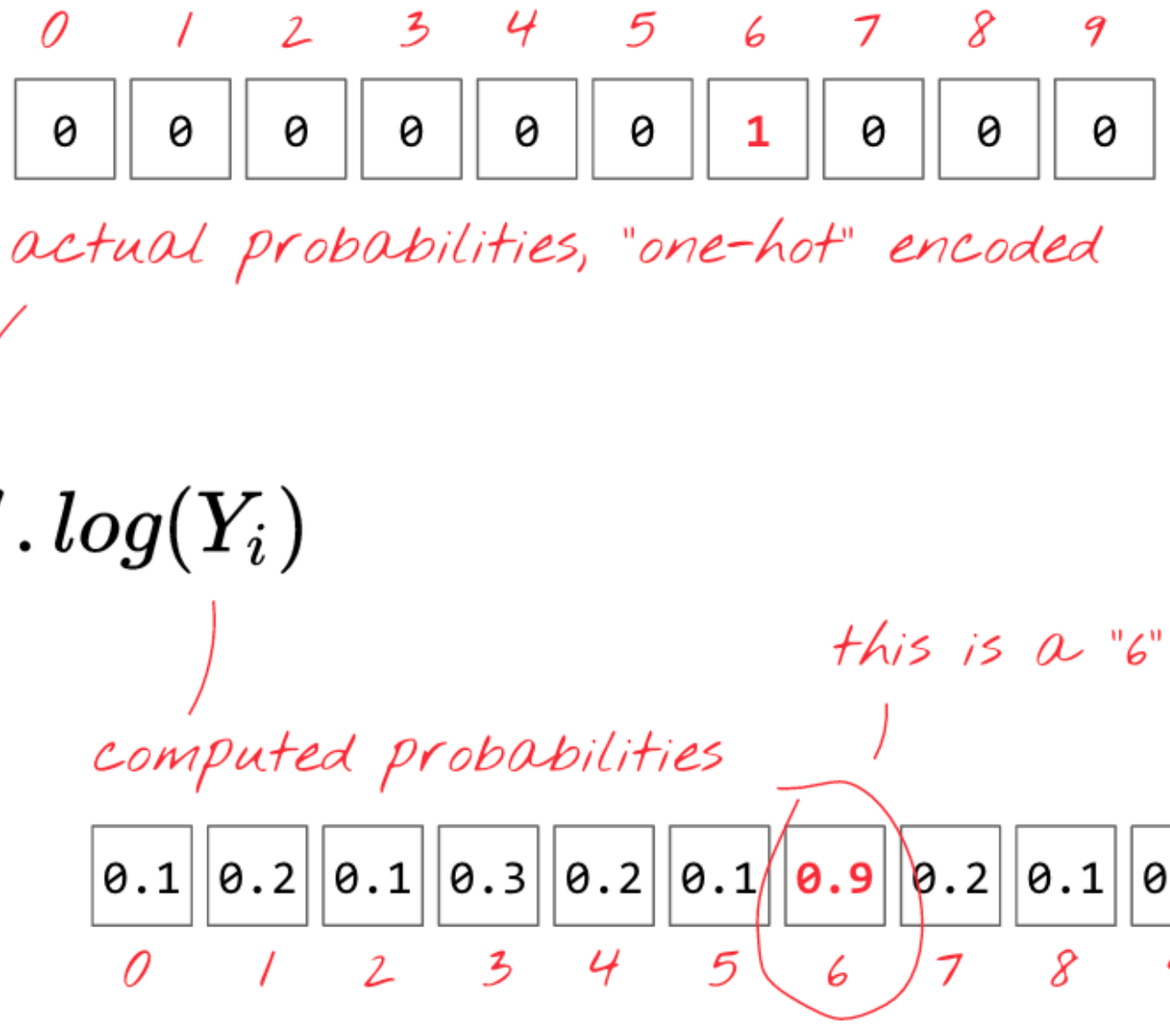
1-LAYER NEURAL NETWORK - ACTIVATION FUNCTION

$$\psi = softmax(L_n) = \frac{e^{L_n}}{||e^L||}$$

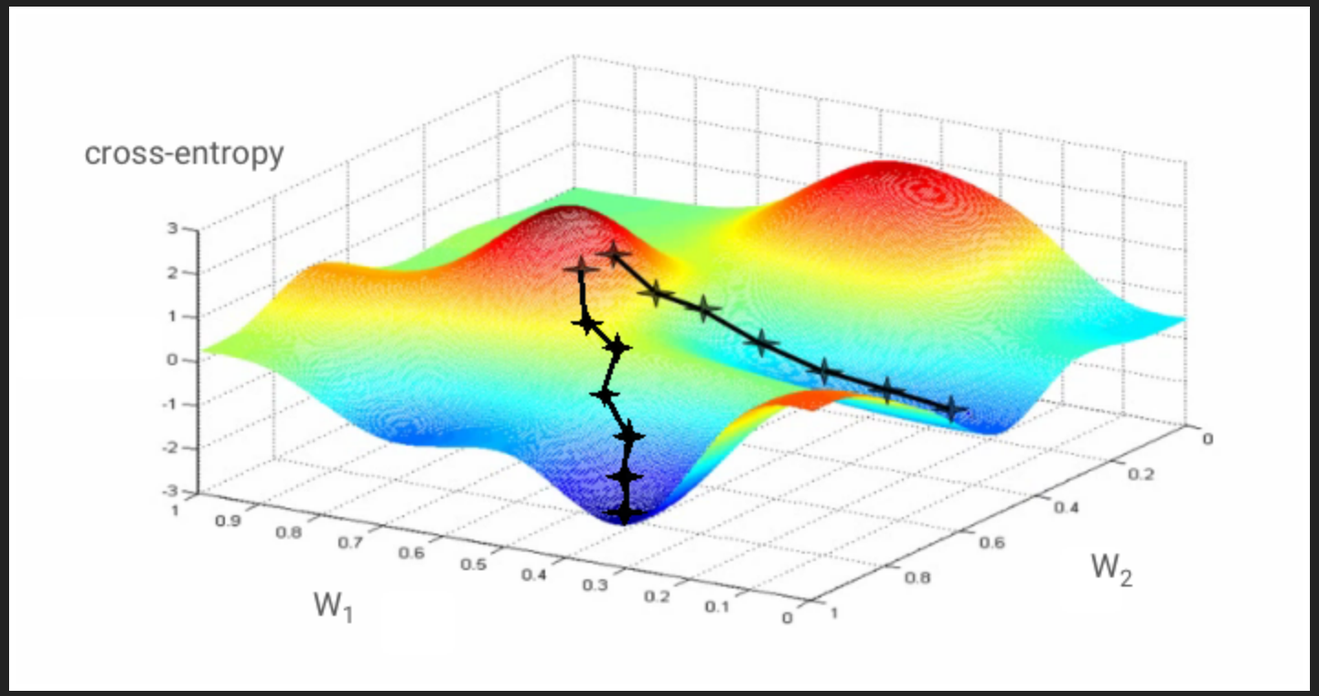
1- LAYER NEURAL NETWORK - FORMULA



1- LAYER NEURAL NETWORK - LOSS FUNCTION



GRADIENT DESCENT



CODE

```
import tensorflow as tf

X = tf.placeholder(tf.float32, [None, 28, 28, 1])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

init = tf.initialize_all_variables()
```

```
# model
Y = tf.nn.softmax(tf.matmul(tf.reshape(X, [-1, 784]), W) + b)
# placeholder for correct labels
Y_ = tf.placeholder(tf.float32, [None, 10])

# loss function
cross_entropy = -tf.reduce_sum(Y_ * tf.log(Y))

# % of correct answers found in batch
is_correct = tf.equal(tf.argmax(Y,1), tf.argmax(Y_,1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

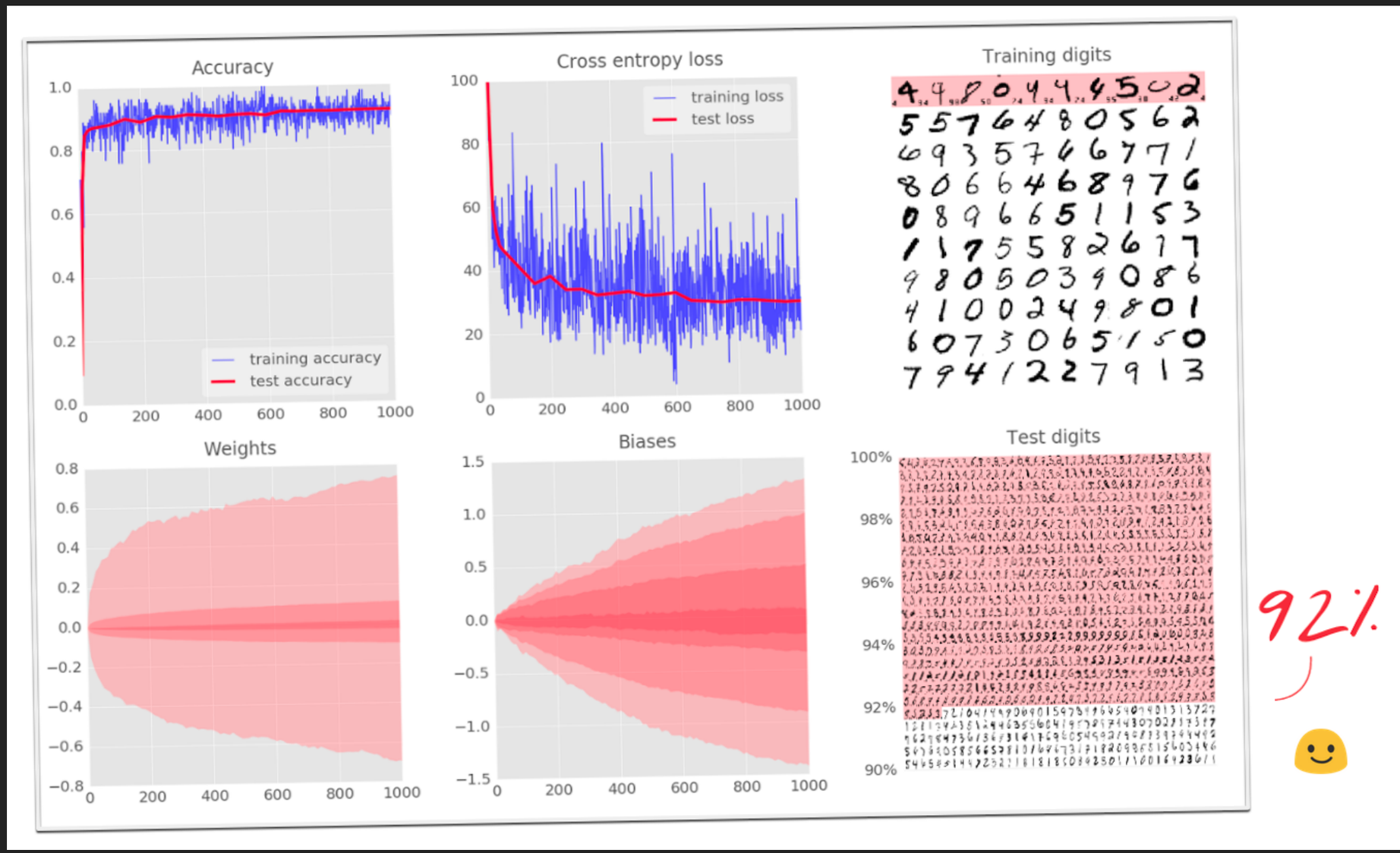
CODE

```
optimizer = tf.train.GradientDescentOptimizer(0.003)
train_step = optimizer.minimize(cross_entropy)
```

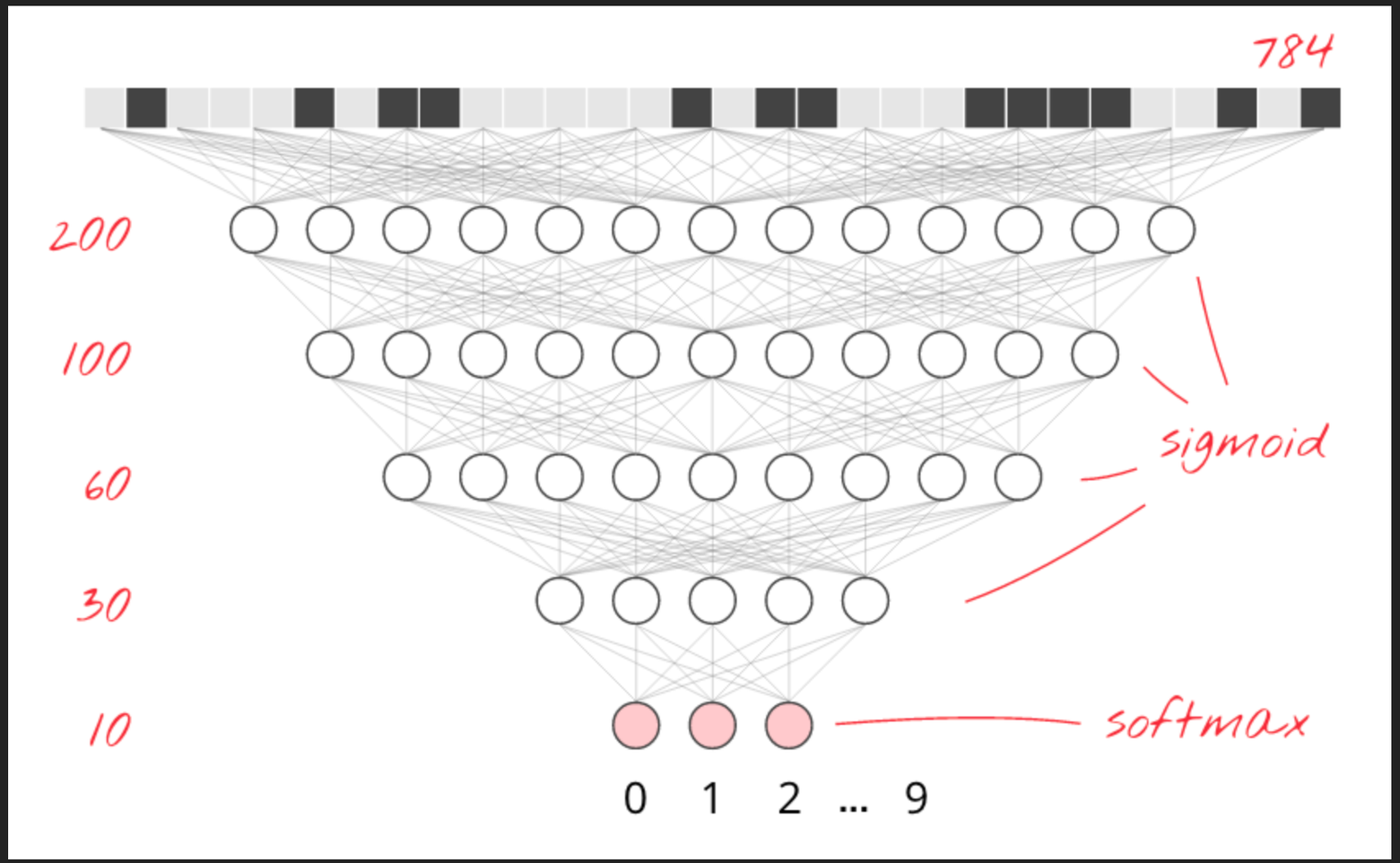
```
sess = tf.Session()
sess.run(init)

for i in range(1000):
    # load batch of images and correct answers
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data={X: batch_X, Y_: batch_Y}
    # train
    sess.run(train_step, feed_dict=train_data)
    # success ?
    a,c = sess.run([accuracy, cross_entropy], feed_dict=train_data)
    # success on test data ?
    test_data={X: mnist.test.images, Y_: mnist.test.labels}
    a,c = sess.run([accuracy, cross_entropy], feed=test_data)
```

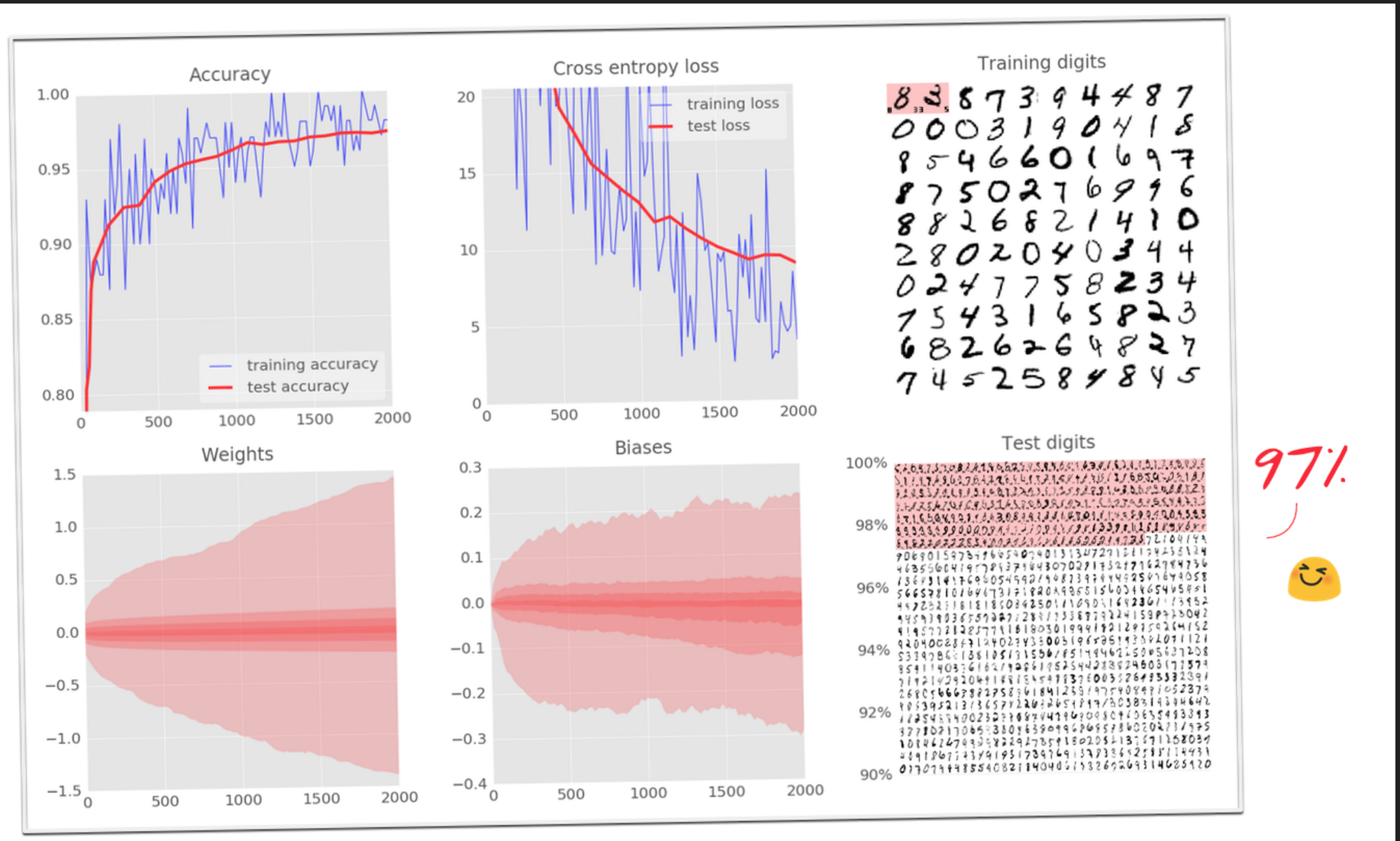
CODE


Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits
92%

DEEP LEARNING



DEEP LEARNING

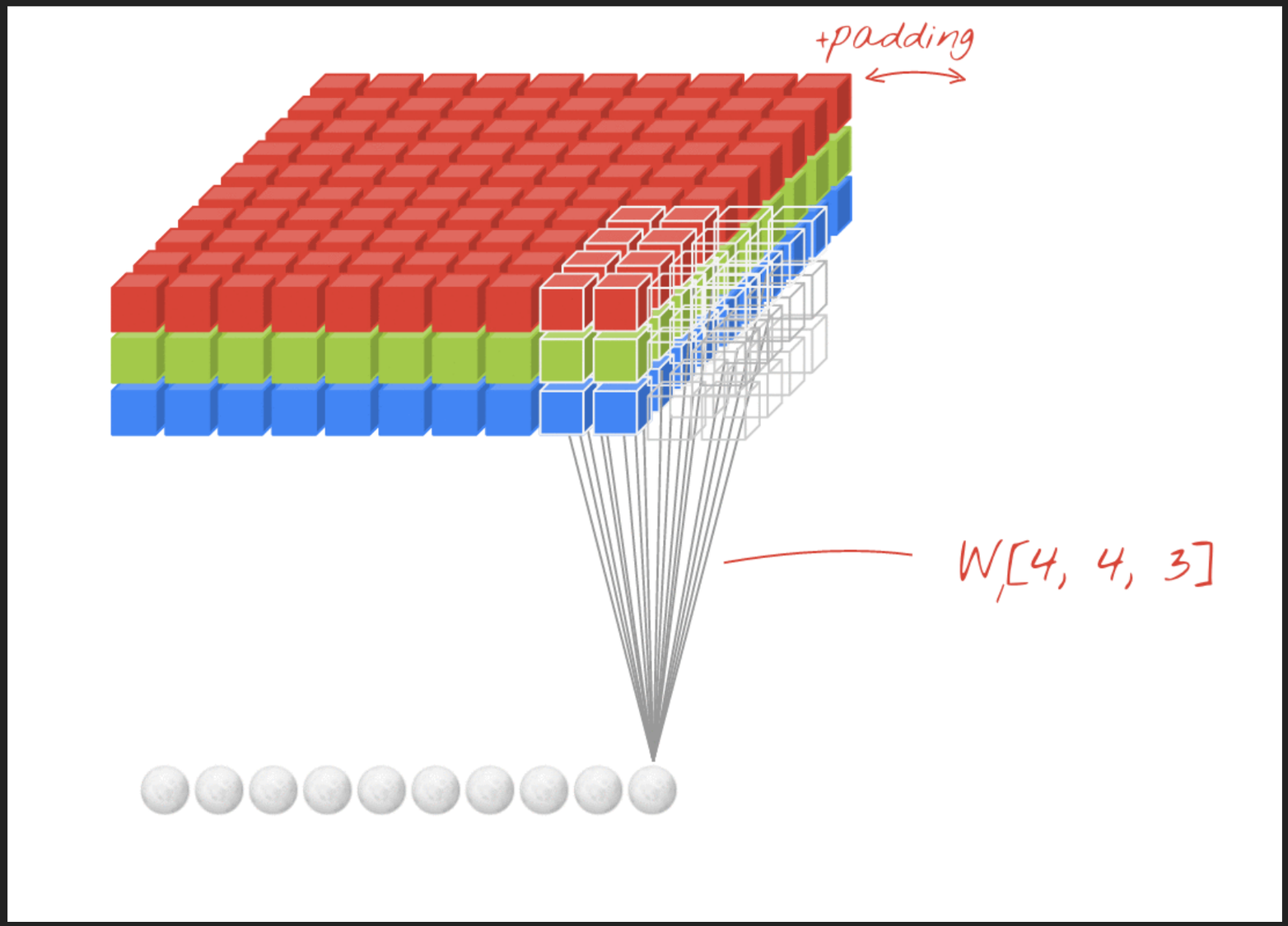


97%

ADVANCED NEURAL NETWORK

- Convolutional networks

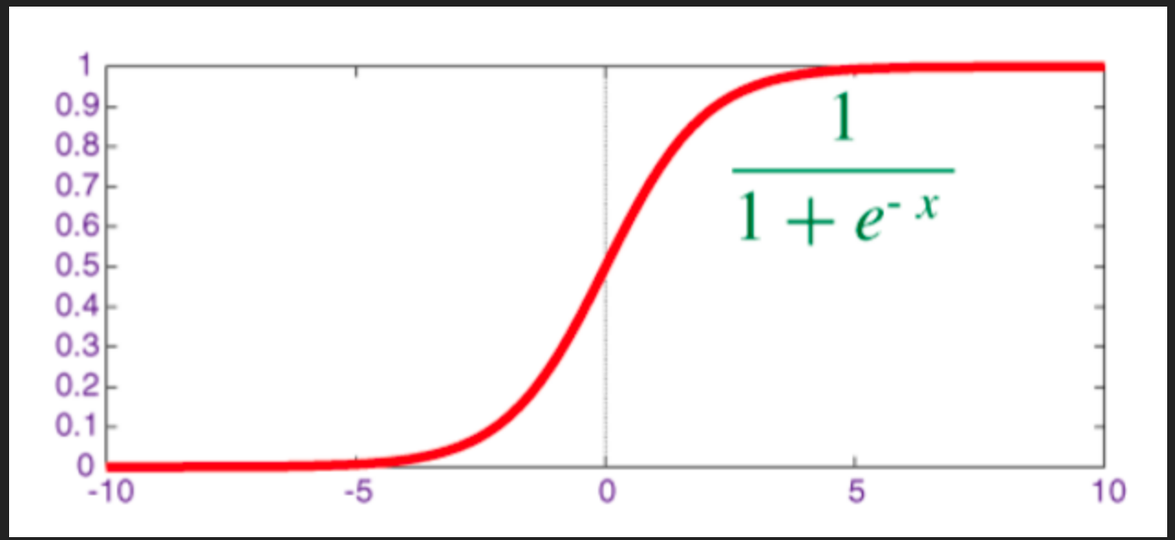
CONVOLUTIONAL NETWORKS



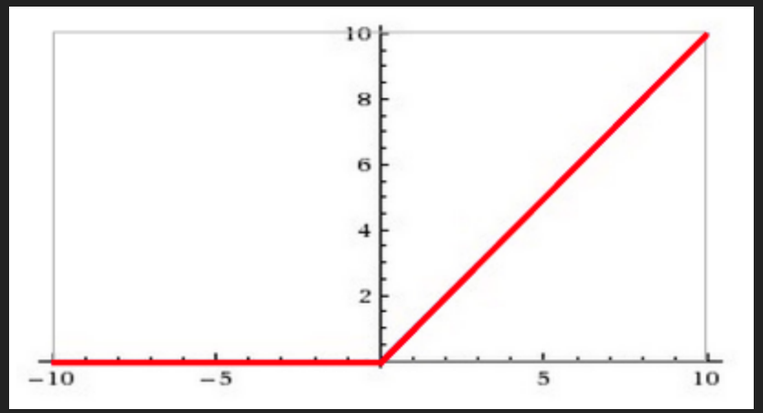
ACTIVATION FUNCTION

- Sigmoid
- Relu

SIGMOID



RELU



REFERENCE

1. TensorFlow and deep learning, without a PhD
2. Deep Learning