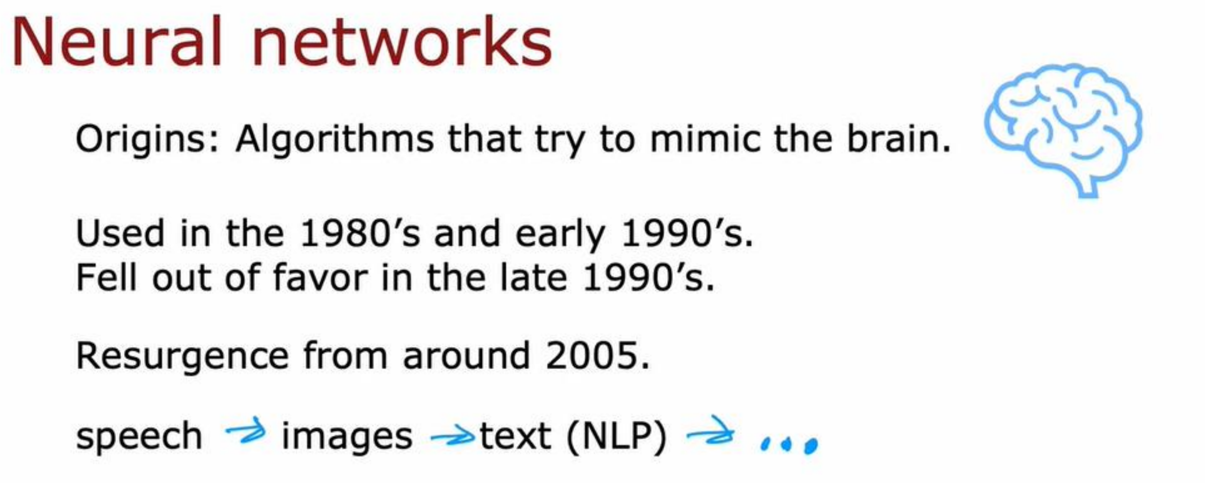
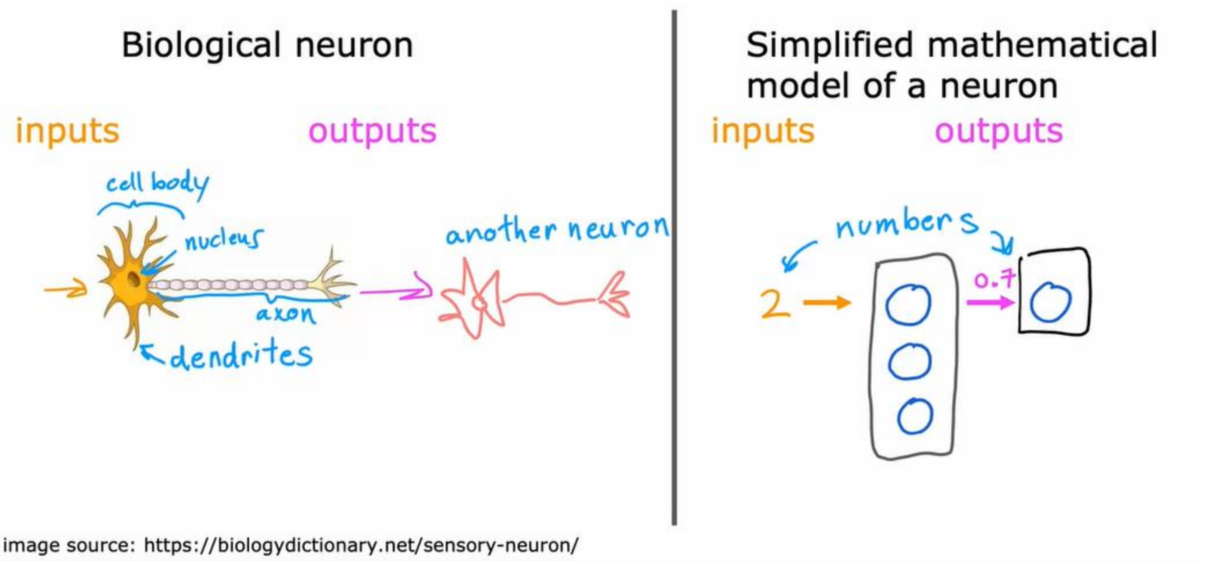
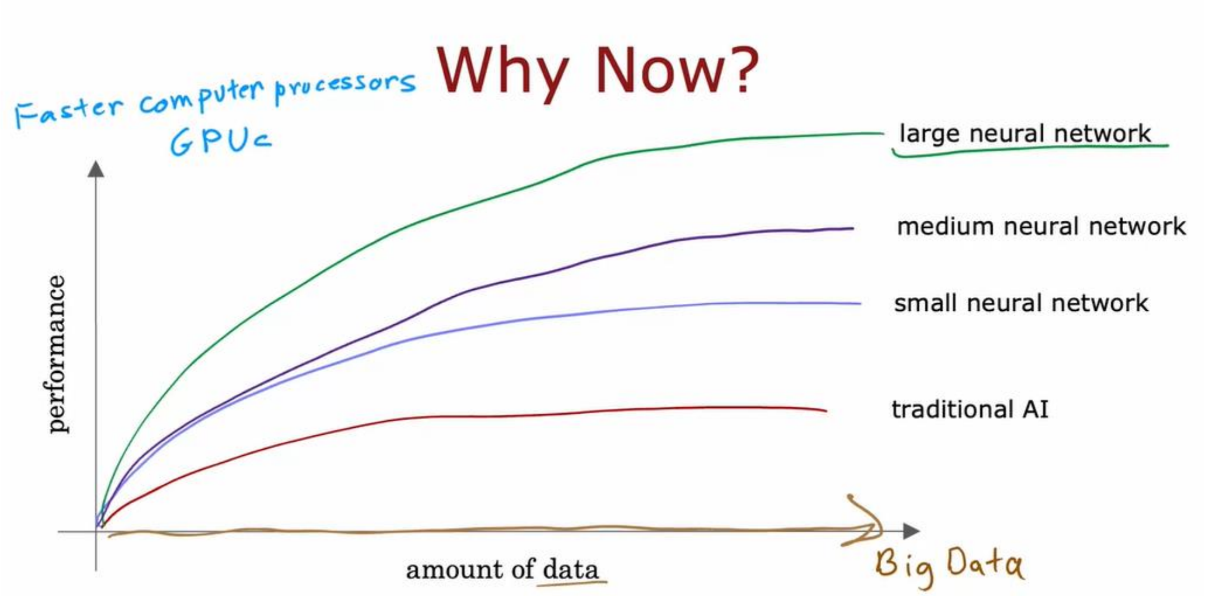
Neural Networks Intuition

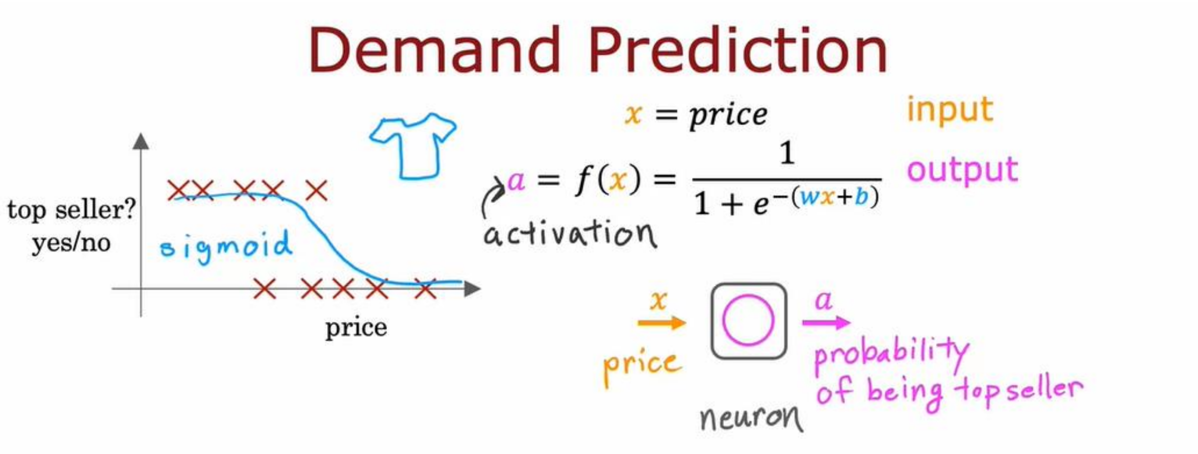
1. Neurons and the brain

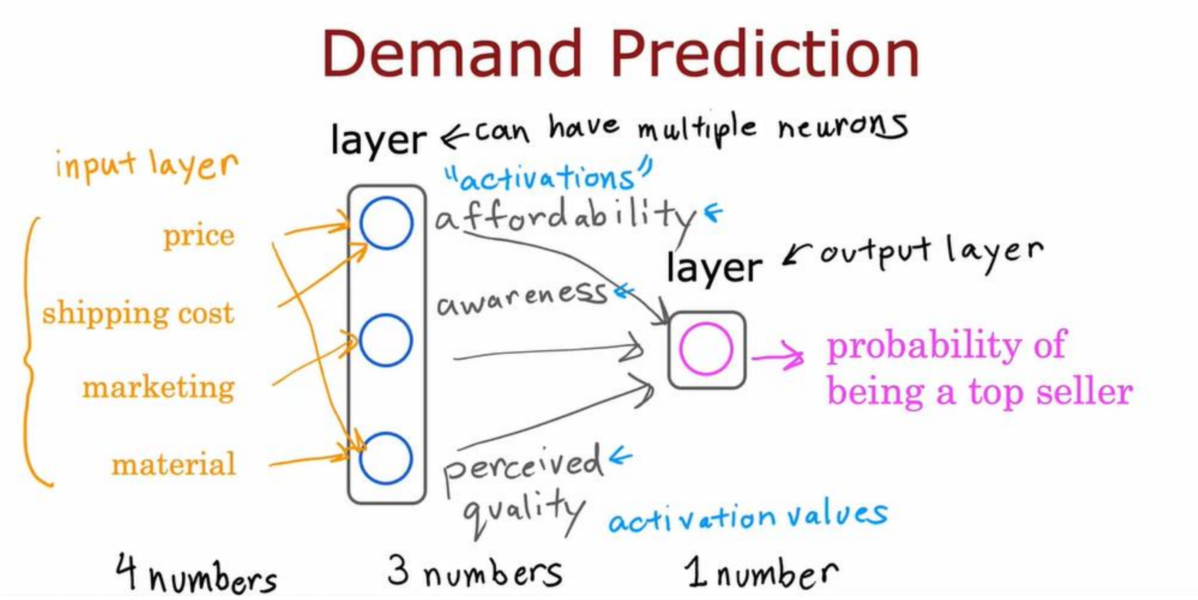


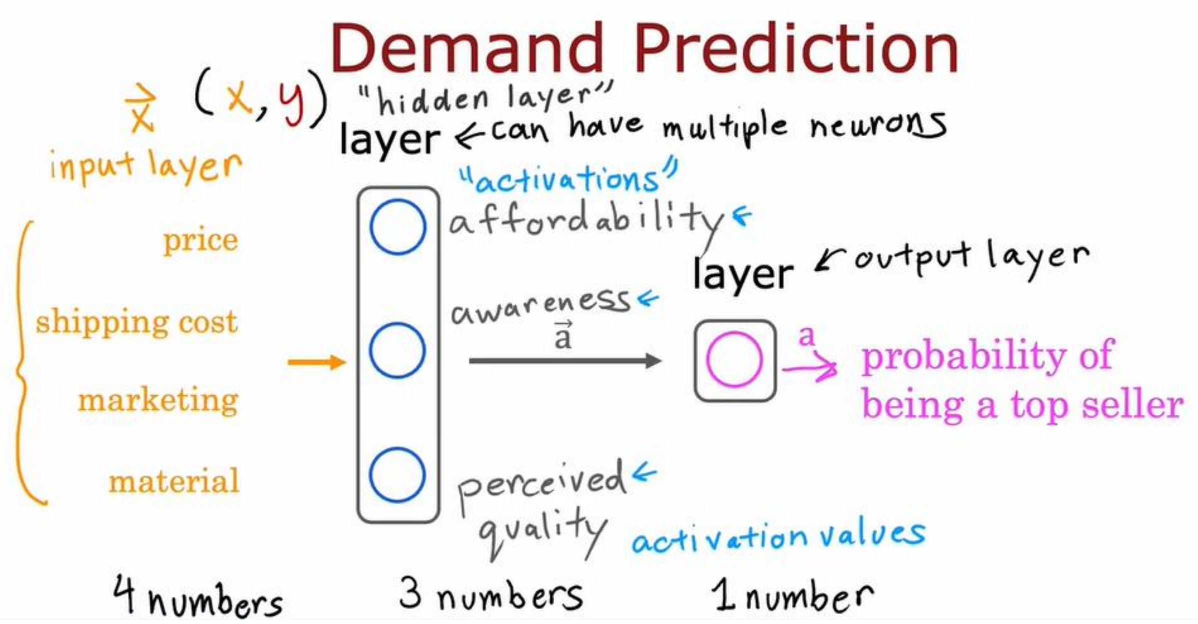


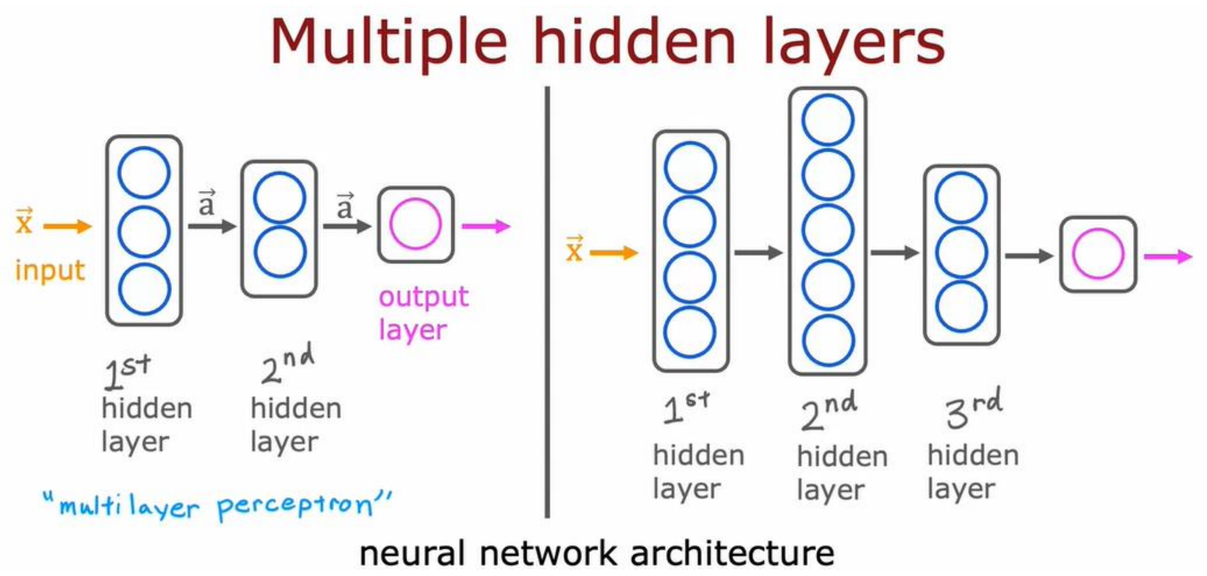


2. Demand prediction

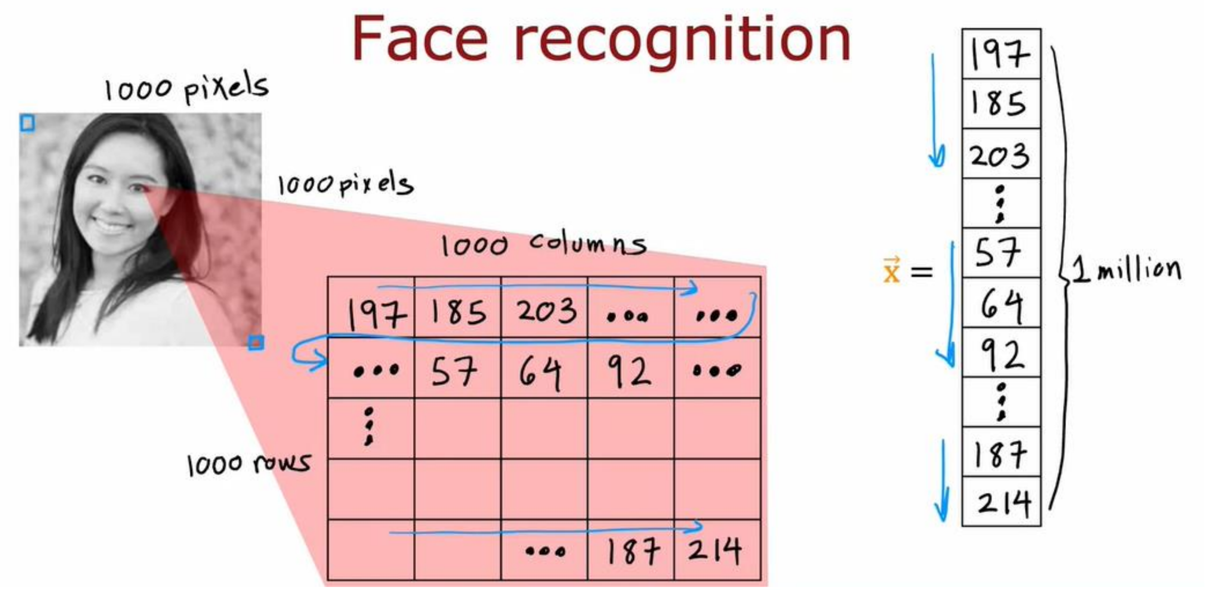


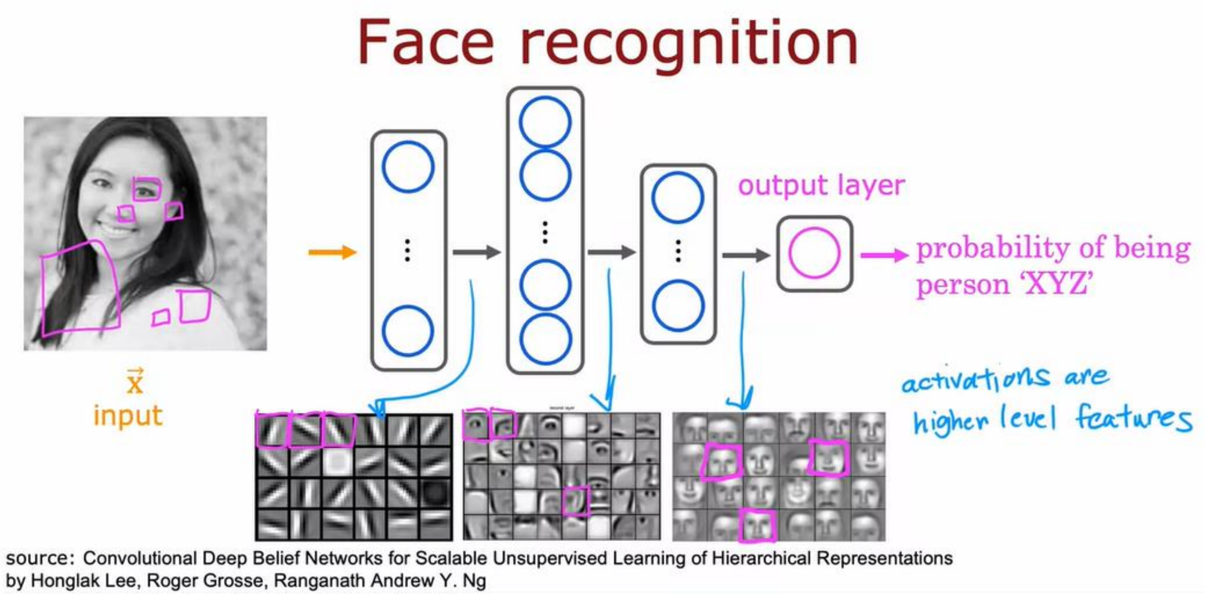


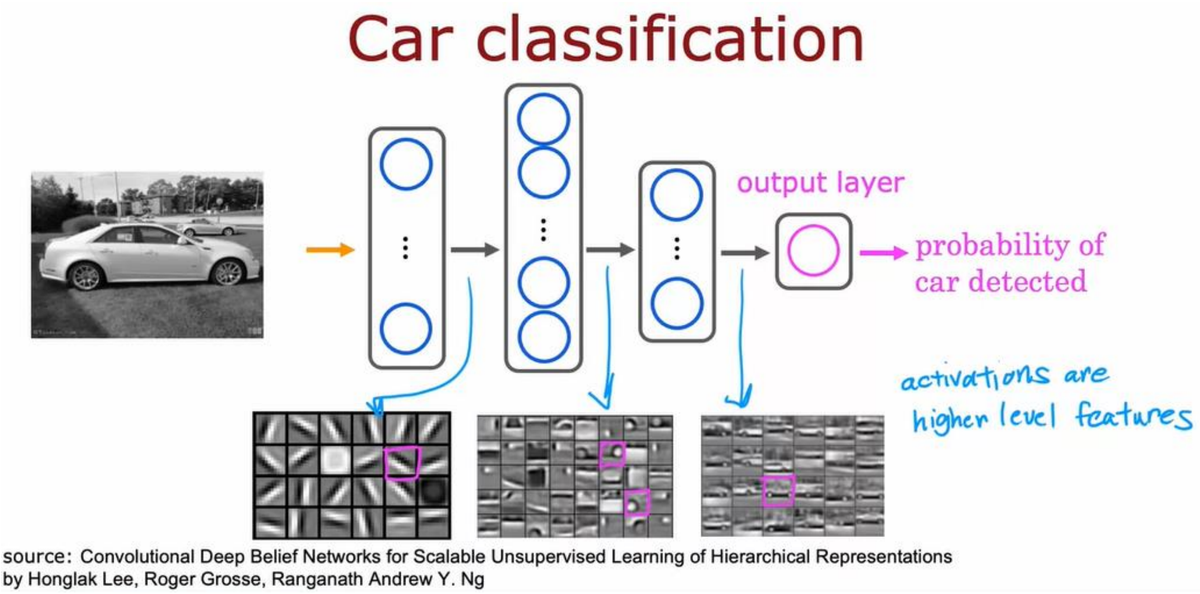




3. Example: recognizing images







4. Practice quiz

Which of these are terms used to refer to components of an artificial neural network? (hint: three of these are correct)

- [x] activation function

> **Correct**
> Yes, an activation is the number calculated by a neuron (and "activations" in the figure above is a vector that is output by a layer that contains multiple neurons).

- [x] neurons

> **Correct**
> Yes, a neuron is a part of a neural network

- [x] layers

> **Correct**
> Yes, a layer is a grouping of neurons in a neural network

- [ ] axon

2. True/False? Neural networks take inspiration from, but do not very accurately mimic, how neurons in a biological brain learn.

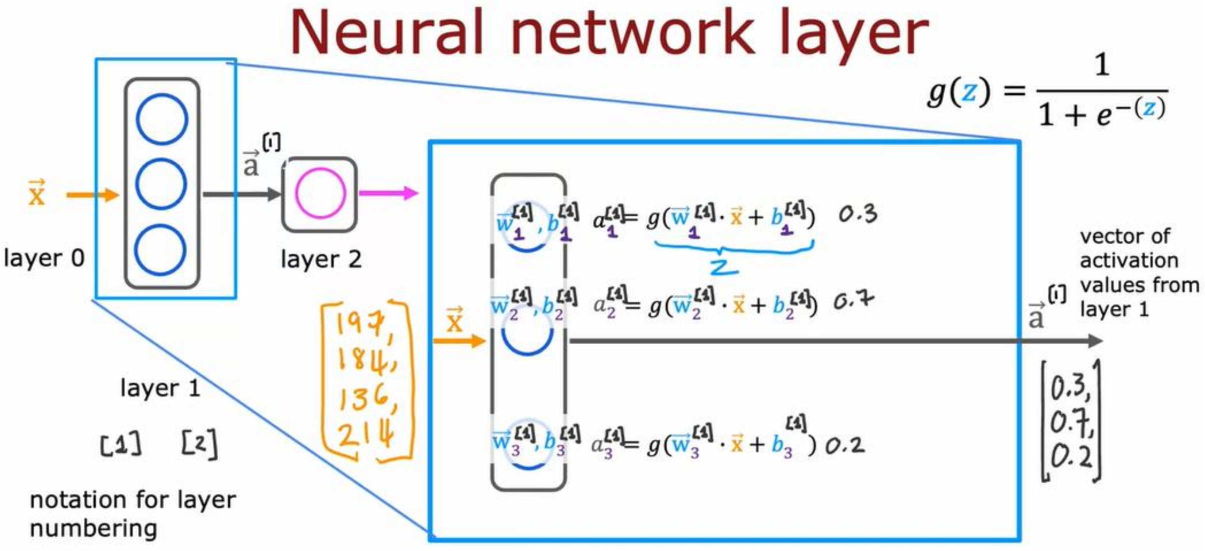
   ○ False

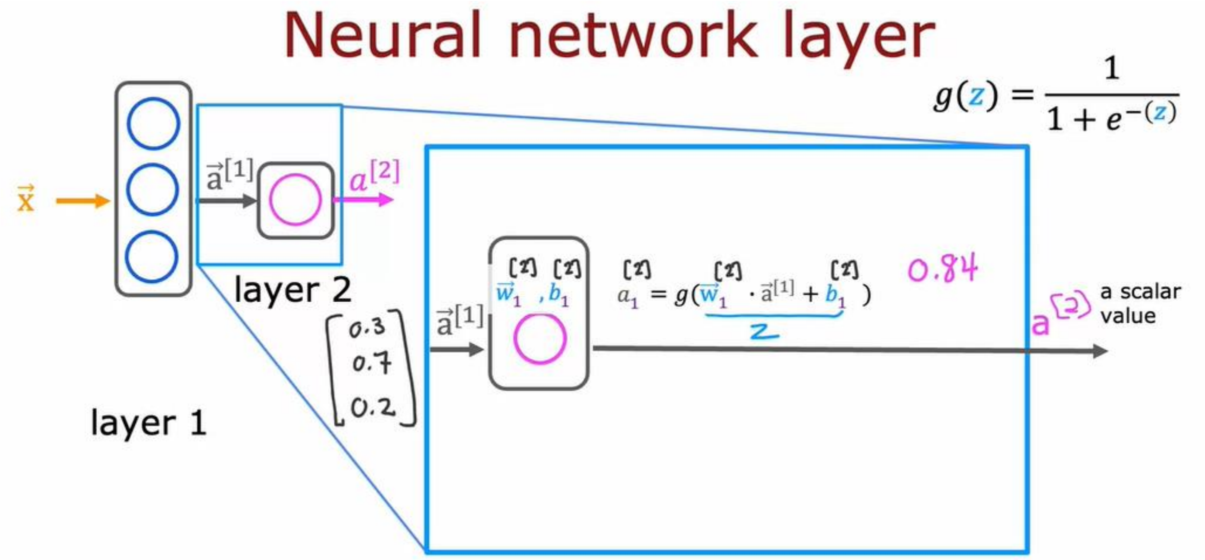
   ◉ True

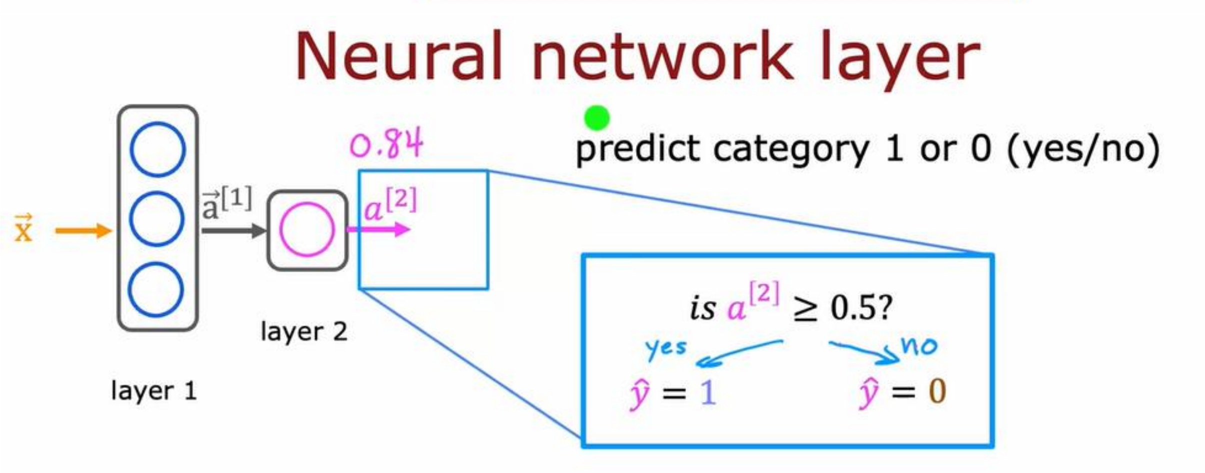
> **Correct**
> Artificial neural networks use a very simplified mathematical model of what a biological neuron does.

Neural network model

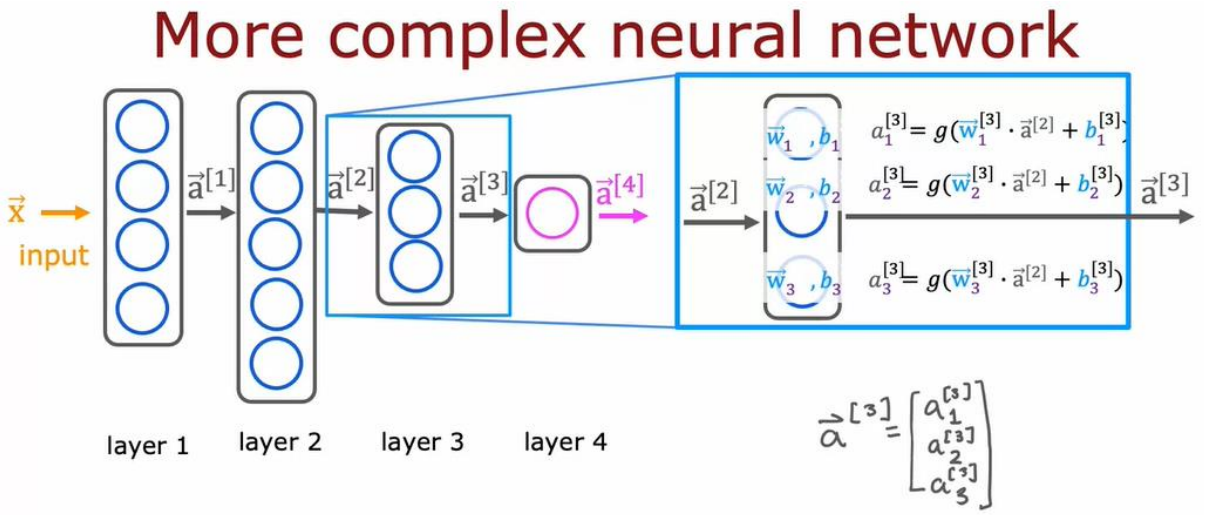
1. Neural network layer

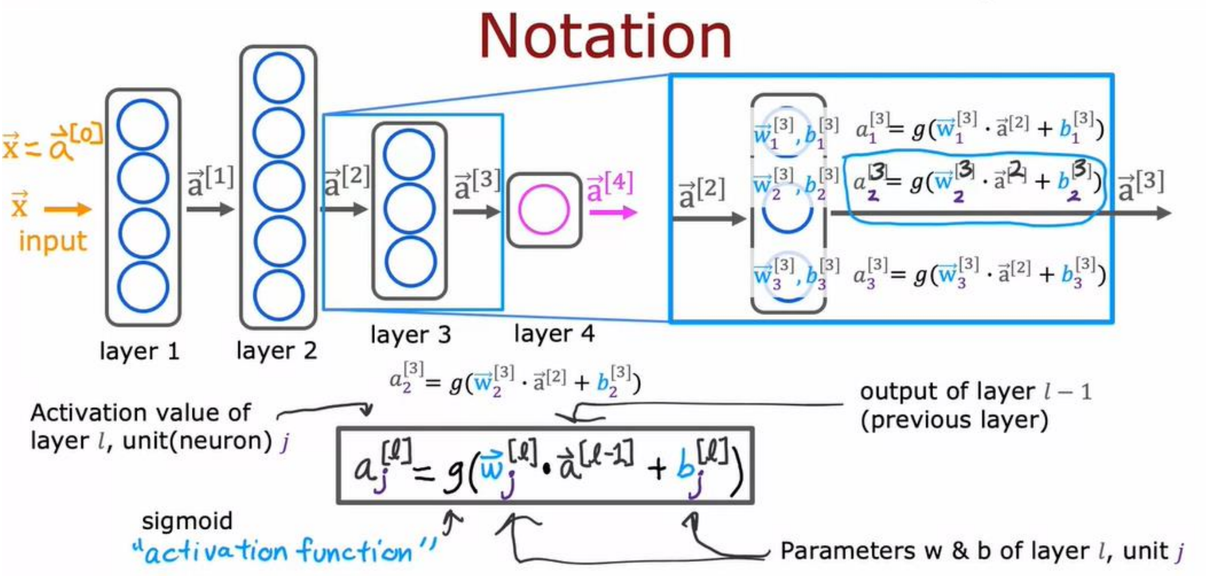




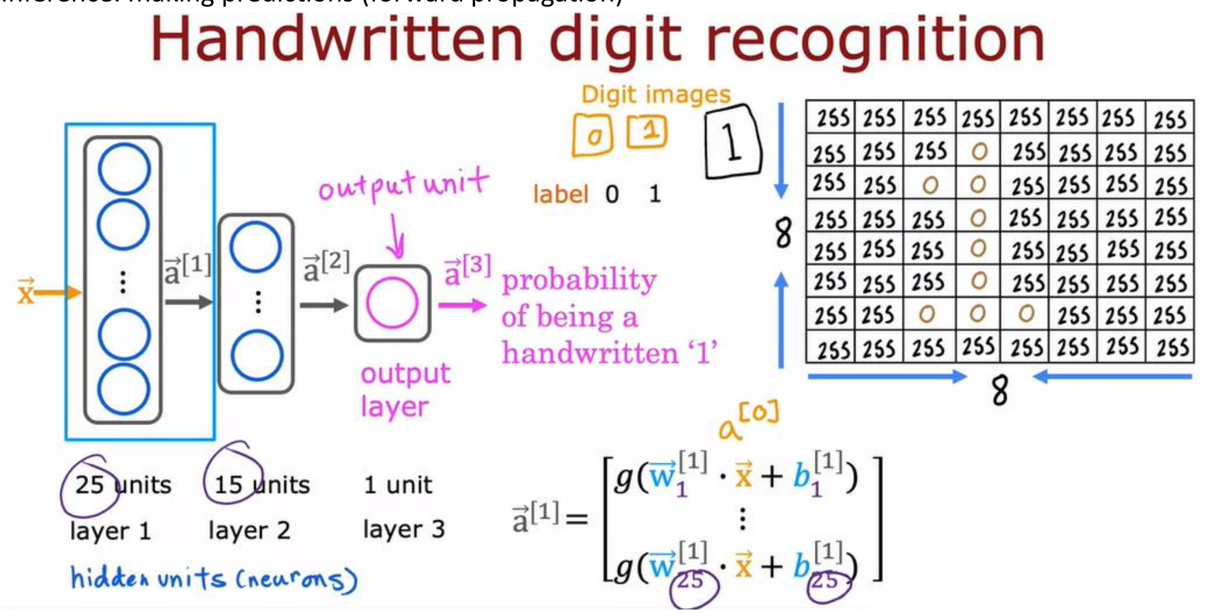


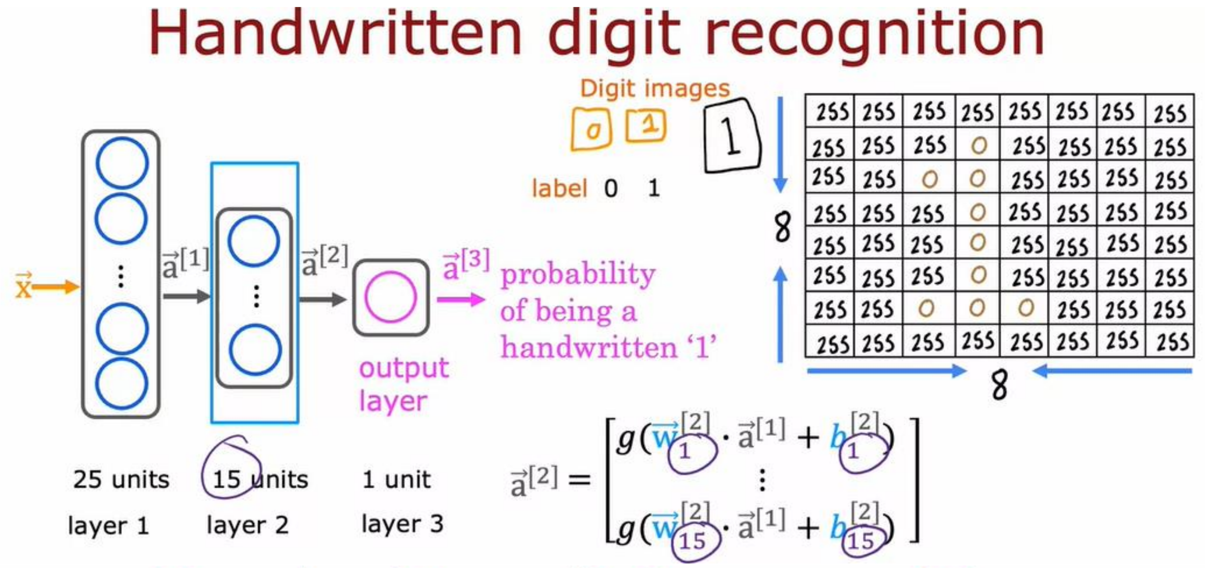
2. More complex neural networks

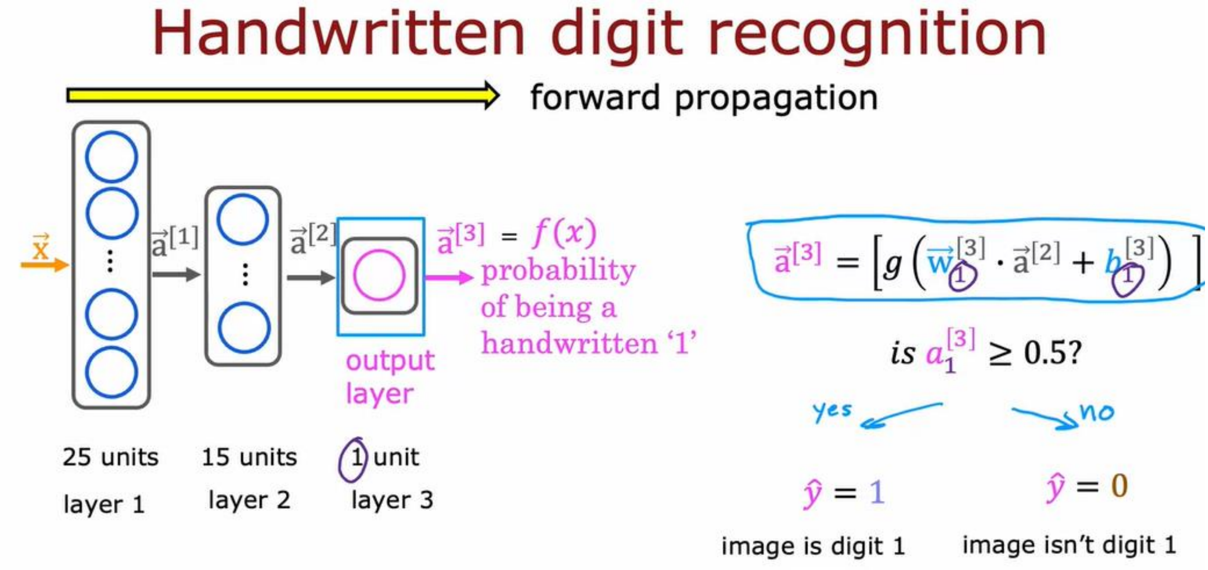




3. Inference: making predictions (forward propagation)







4. Practice quiz

$$\boxed{a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})}$$

For a neural network, what is the expression for calculating the activation of the third neuron in layer 2? Note, thi is different from the question that you saw in the lecture video.

- ○ $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[2]} + b_3^{[2]})$
- ○ $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[1]} + b_2^{[3]})$
- ○ $a_3^{[2]} = g(\vec{w}_2^{[3]} \cdot \vec{a}^{[2]} + b_2^{[3]})$
- ◉ $a_3^{[2]} = g(\vec{w}_3^{[2]} \cdot \vec{a}^{[1]} + b_3^{[2]})$

Correct
Yes! The superscript [2] refers to layer 2. The subscript 3 refers to the neuron in that layer. The input to layer 2 is the activation vector from layer 1.

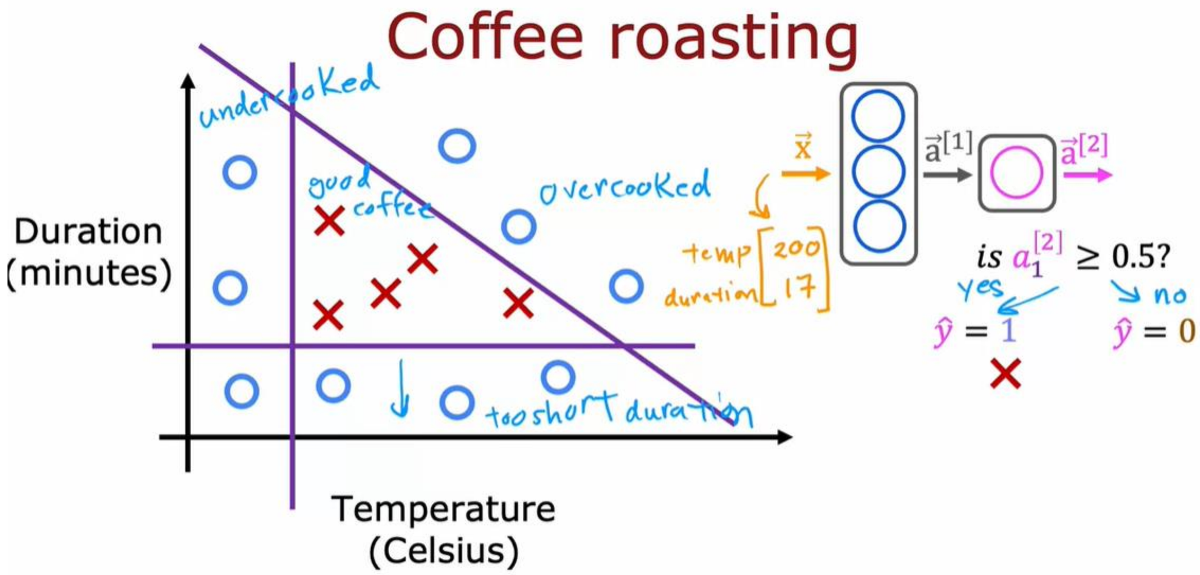
For the handwriting recognition task discussed in lecture, what is the output $a_1^{[3]}$?

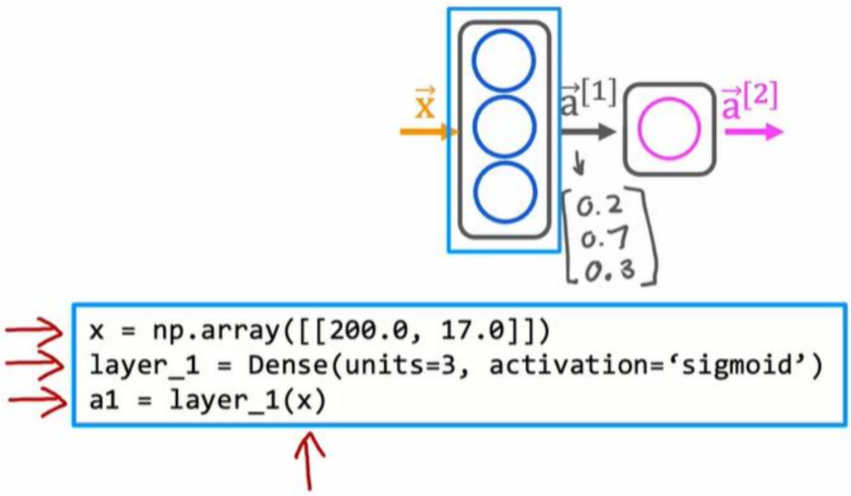
- ○ A vector of several numbers that take values between 0 and 1
- ○ A vector of several numbers, each of which is either exactly 0 or 1
- ○ A number that is either exactly 0 or 1, comprising the network's prediction
- ◉ The estimated probability that the input image is of a number 1, a number that ranges from 0 to 1.

Correct
Yes! The neural network outputs a single number between 0 and 1.

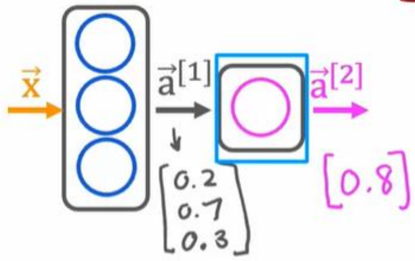
TensorFlow implementation

1. Inference in code





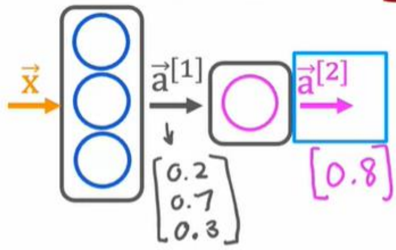
Build the model using TensorFlow




```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

```
layer_2 = Dense(units=1, activation='sigmoid')
a2 = layer_2(a1)
```

Build the model using TensorFlow



is $a_1^{[2]} \geq 0.5$?

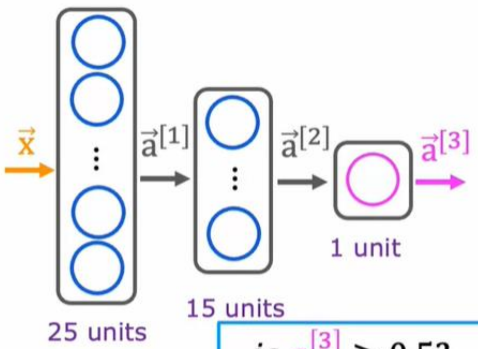
yes → $\hat{y} = 1$ no → $\hat{y} = 0$

```
x = np.array([[200.0, 17.0]])
layer_1 = Dense(units=3, activation='sigmoid')
a1 = layer_1(x)
```

```
if a2 >= 0.5:
  yhat = 1
else:
  yhat = 0
```

```
layer_2 = Dense(units=1, activation='sigmoid')
a2 = layer_2(a1)
```

Model for digit classification




```
x = np.array([[0.0,...245,...240...0]])
layer_1 = Dense(units=25, activation='sigmoid')
a1 = layer_1(x)

layer_2 = Dense(units=15, activation='sigmoid')
a2 = layer_2(a1)

layer_3 = Dense(units=1, activation='sigmoid')
a3 = layer_3(a2)
```

is $a_1^{[3]} \geq 0.5$?

$\hat{y} = 1$ $\hat{y} = 0$

```
if a3 >= 0.5:
  yhat = 1
else:
  yhat = 0
```

2. Data in TensorFlow

Feature vectors

temperature (Celsius)	duration (minutes)	Good coffee? (1/0)
200.0	17.0	1
425.0	18.5	0
...	...	...

```
x = np.array([[200.0, 17.0]])
[[200.0, 17.0]]
```

why?

Note about numpy arrays

3 columns

2 rows

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

2 x 3 matrix

```
x = np.array([[1, 2, 3],
              [4, 5, 6]])
[[1, 2, 3],
 [4, 5, 6]]
```

2D array

2 x 3

4 rows

$$\begin{bmatrix} 0.1 & 0.2 \\ -3 & -4 \\ -.5 & -.6 \\ 7 & 8 \end{bmatrix}$$

2 columns

4 x 2 matrix

```
x = np.array([[0.1, 0.2],
              [-3.0, -4.0,],
              [-0.5, -0.6,],
              [7.0, 8.0,]])
[[0.1, 0.2],
 [-3.0, -4.0,],
 [-0.5, -0.6,],
 [7.0, 8.0,]]
```

4 x 2

1 x 2

2 x 1

Note about numpy arrays

```
x = np.array([[200, 17]])
```

$$\begin{bmatrix} 200 & 17 \end{bmatrix}$$

1 x 2

```
x = np.array([[200],
              [17]])
```

$$\begin{bmatrix} 200 \\ 17 \end{bmatrix}$$

2 x 1

```
x = np.array([200,17])
```

1D "Vector"

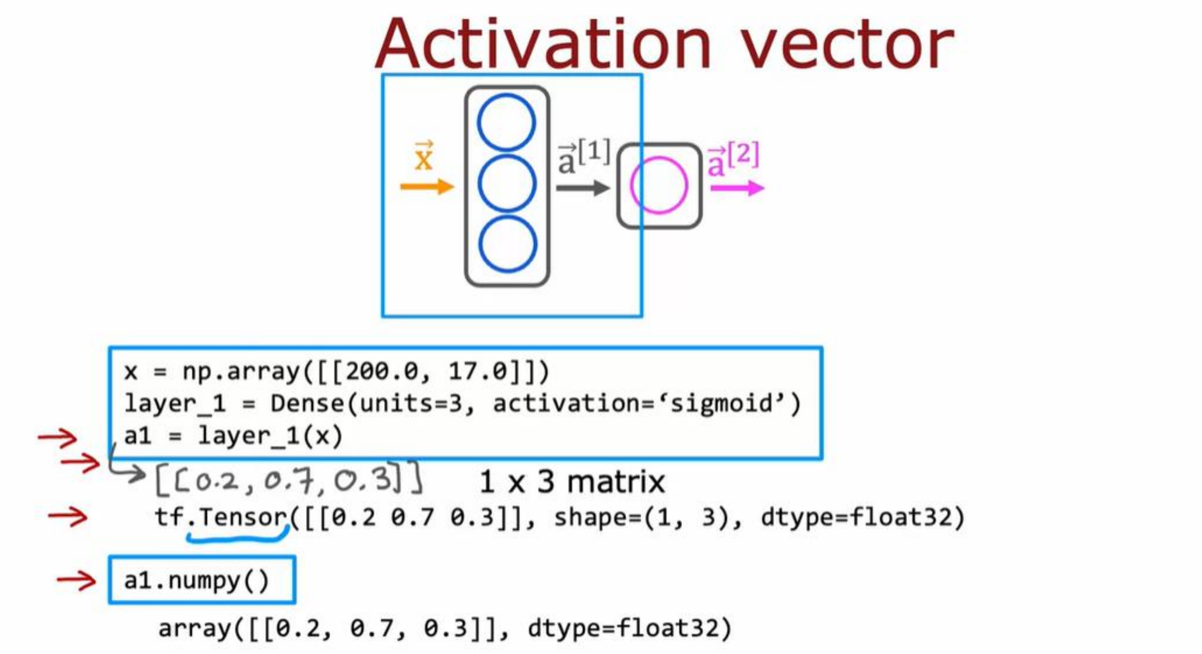
Feature vectors

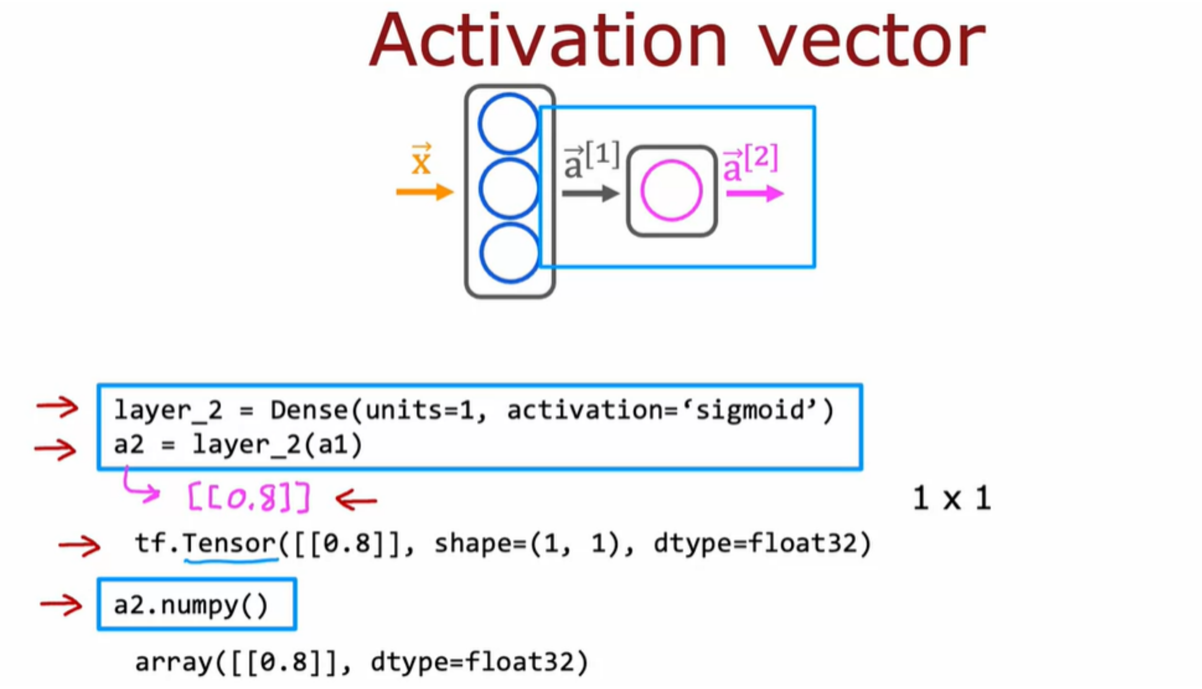
temperature (Celsius)	duration (minutes)	Good coffee? (1/0)
200.0	17.0	1
425.0	18.5	0
...	...	...

```
x = np.array([[200.0, 17.0]])
[[200.0, 17.0]]
```

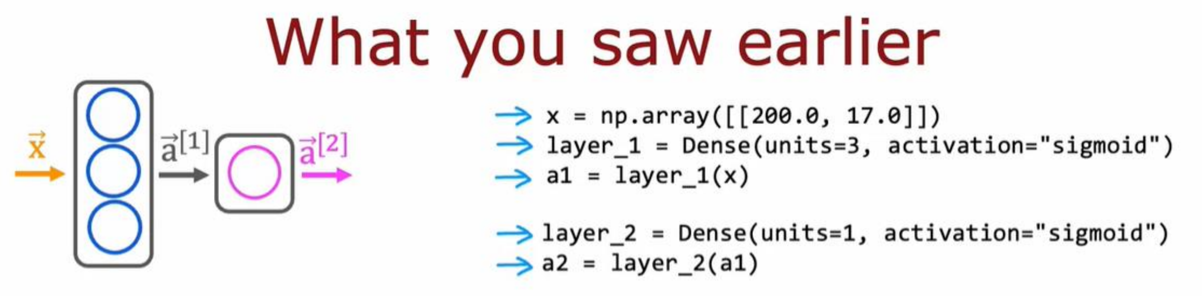
1 x 2

$$\begin{bmatrix} 200.0 & 17.0 \end{bmatrix}$$

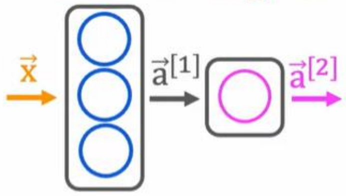




3. Building a neural network



Building a neural network architecture




```
layer_1 = Dense(units=3, activation="sigmoid")
layer_2 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2])
```

layer1 layer2

		y
200	17	1
120	5	0
425	20	0
212	18	1

```
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])
```

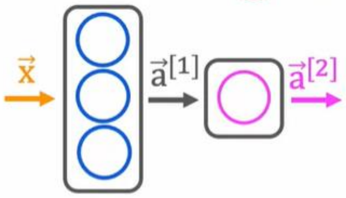
4 x 2

targets

```
y = np.array([1,0,0,1])
model.compile(...)
model.fit(x,y)
model.predict(x_new)
```

more about this next week!

Building a neural network architecture




```
model = Sequential([
  Dense(units=3, activation="sigmoid"),
  Dense(units=1, activation="sigmoid")])
```

		y
200	17	1
120	5	0
425	20	0
212	18	1

```
x = np.array([[200.0, 17.0],
              [120.0, 5.0],
              [425.0, 20.0],
              [212.0, 18.0]])
```

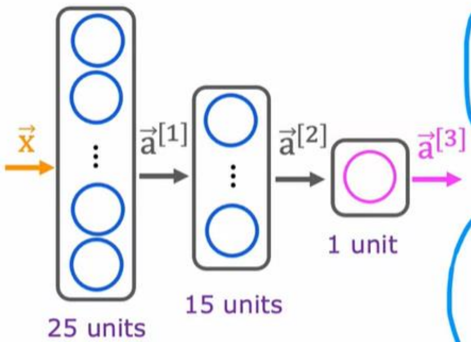
4 x 2

targets

```
y = np.array([1,0,0,1])
model.compile(...)
model.fit(x,y)
model.predict(x_new)
```

more about this next week!

Digit classification model




```
layer_1 = Dense(units=25, activation="sigmoid")
layer_2 = Dense(units=15, activation="sigmoid")
layer_3 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2, layer_3])
model.compile(...)
x = np.array([[0..., 245, ..., 17],
              [0..., 200, ..., 184]])
y = np.array([1,0])
model.fit(x,y)
model.predict(x_new)
```

more about this next week!

4. Practice quiz

1. For the the following code:

```
model = Sequential([
 Dense(units=25, activation="sigmoid"),
 Dense(units=15, activation="sigmoid"),
 Dense(units=10, activation="sigmoid"),
 Dense(units=1, activation="sigmoid")])
```

This code will define a neural network with how many layers?

- ○ 5
- ○ 3
- ◉ 4
- ○ 25

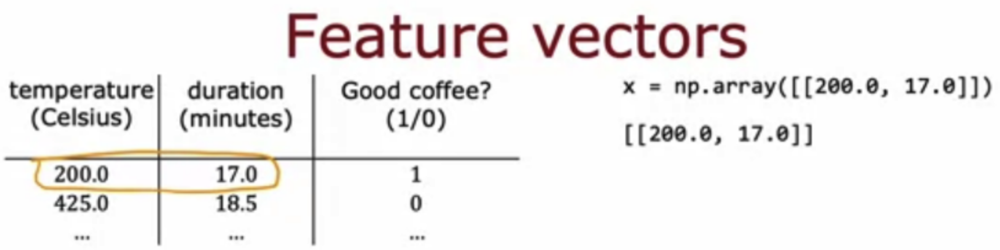
Correct
Yes! Each call to the "Dense" function defines a layer of the neural network.

How do you define the second layer of a neural network that has 4 neurons and a sigmoid activation?

- ○ Dense(units=[4], activation=['sigmoid'])
- ○ Dense(layer=2, units=4, activation = 'sigmoid')
- ◉ Dense(units=4, activation='sigmoid')
- ○ Dense(units=4)

Correct
Yes! This will have 4 neurons and a sigmoid activation.

3.



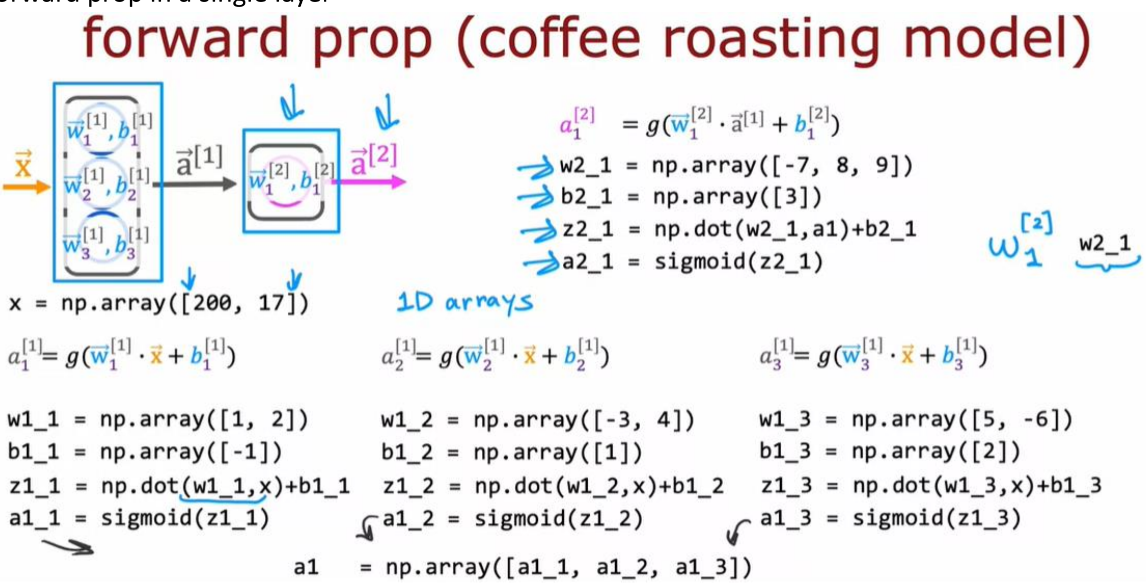
If the input features are temperature (in Celsius) and duration (in minutes), how do you write the code for the first feature vector x shown above?

- ○ x = np.array([['200.0', '17.0']])
- ○ x = np.array([[200.0],[17.0]])
- ◉ x = np.array([[200.0, 17.0]])
- ○ x = np.array([[200.0 + 17.0]])

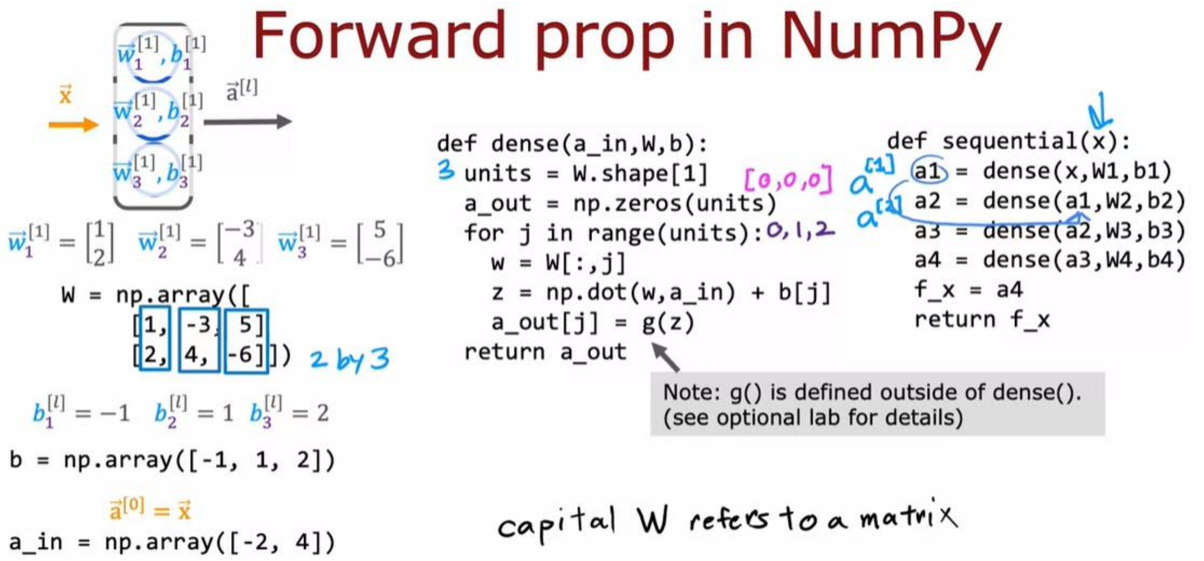
Correct
Yes! A row contains all the features of a training example. Each column is a feature.

Neural network implementation in python

1. Forward prop in a single layer



2. General implementation of forward prop



3. Practice quiz

According to the lecture, how do you calculate the activation of the third neuron in the first layer using NumPy?

- (selected)

  z1_3 = np.dot(w1_3, x) + b1_3

  a1_3 = sigmoid(z1_3)

- z1_3 =w1_3 * x + b

  a1_3 = sigmoid(z1_3)

- layer_1 = Dense(units=3, activation='sigmoid')

  a_1 = layer_1(x)

Correct

Correct. Use the numpy.dot function to take the dot product. The sigmoid function shown in lecture can be a function that you write yourself (see course 1, week 3 of this specialization), and that will be provided to you in this course.

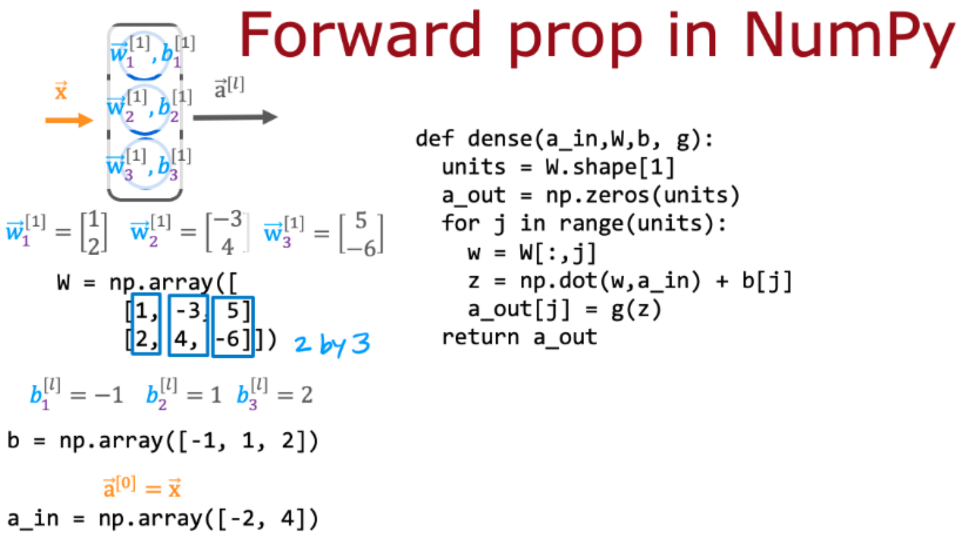
According to the lecture, when coding up the numpy array W, where would you place the w parameters for each neuron?

- (selected) In the columns of W.
- In the rows of W.

Correct

Correct. The w parameters of neuron 1 are in column 1. The w parameters of neuron 2 are in column 2, and so on.

3.



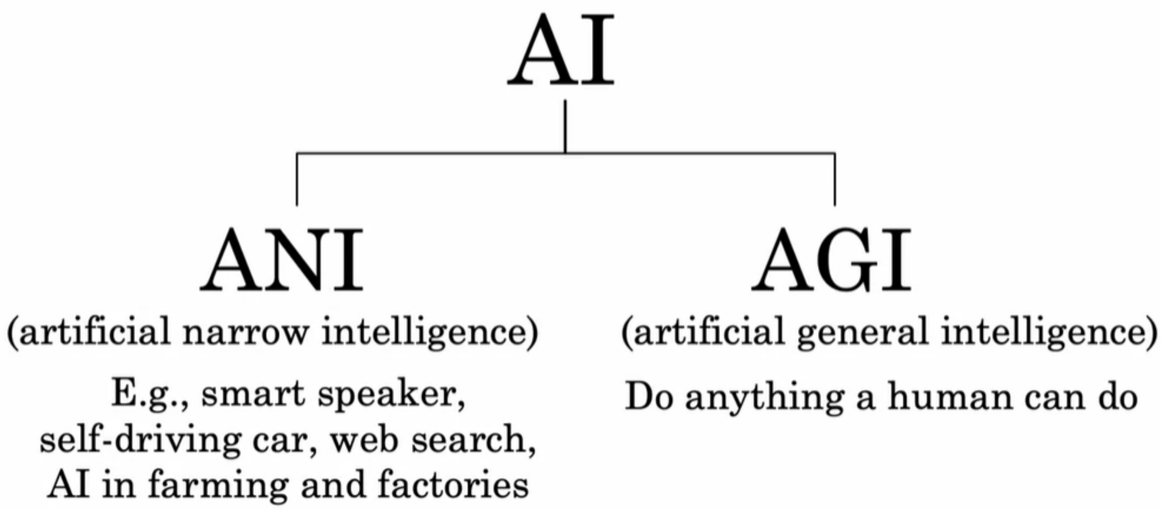
For the code above in the "dense" function that defines a single layer of neurons, how many times does the code go through the "for loop"? Note that W has 2 rows and 3 columns.

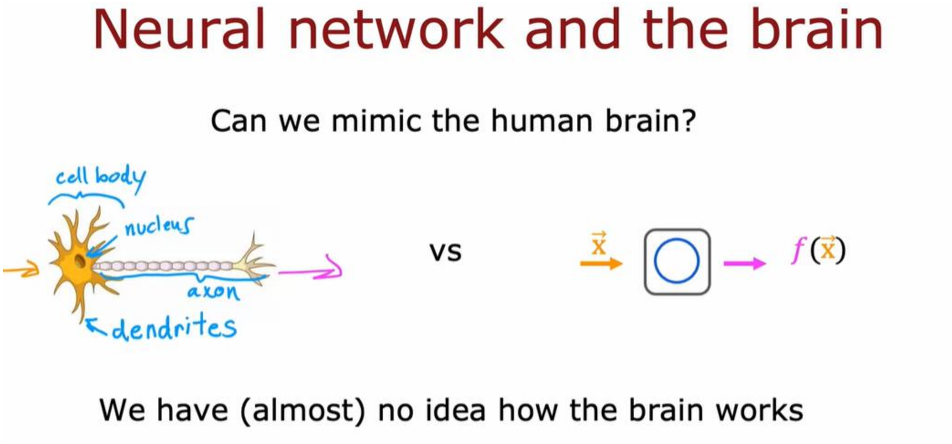
- ○ 2 times
- ◉ 3 times
- ○ 6 times
- ○ 5 times

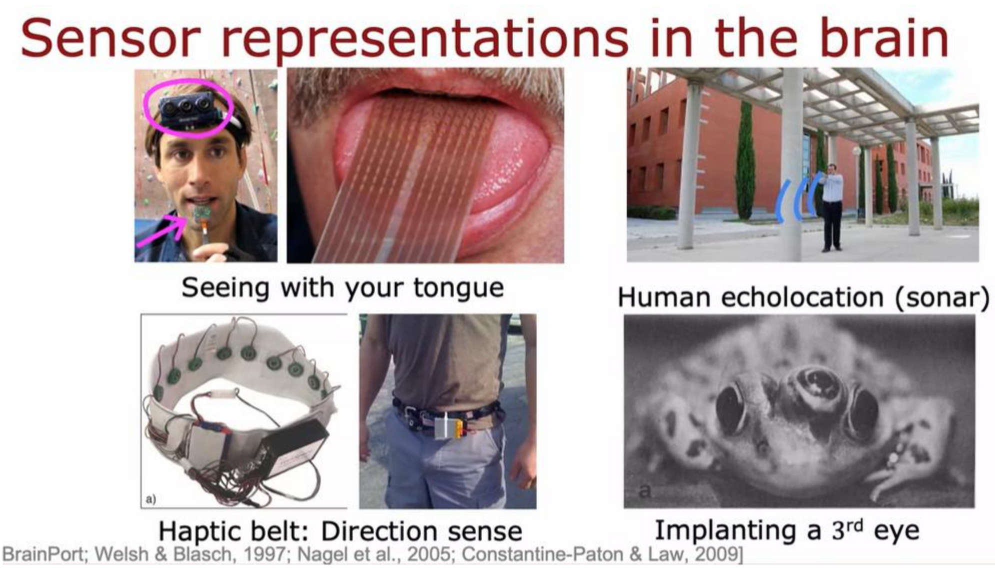
Correct
Yes! For each neuron in the layer, there is one column in the numpy array W. The for loop calculates the activation value for each neuron. So if there are 3 columns in W, there are 3 neurons in the dense layer, and therefore the for loop goes through 3 iterations (one for each neuron).

Speculations on artificial general intelligence

1. Is there a path to AGI

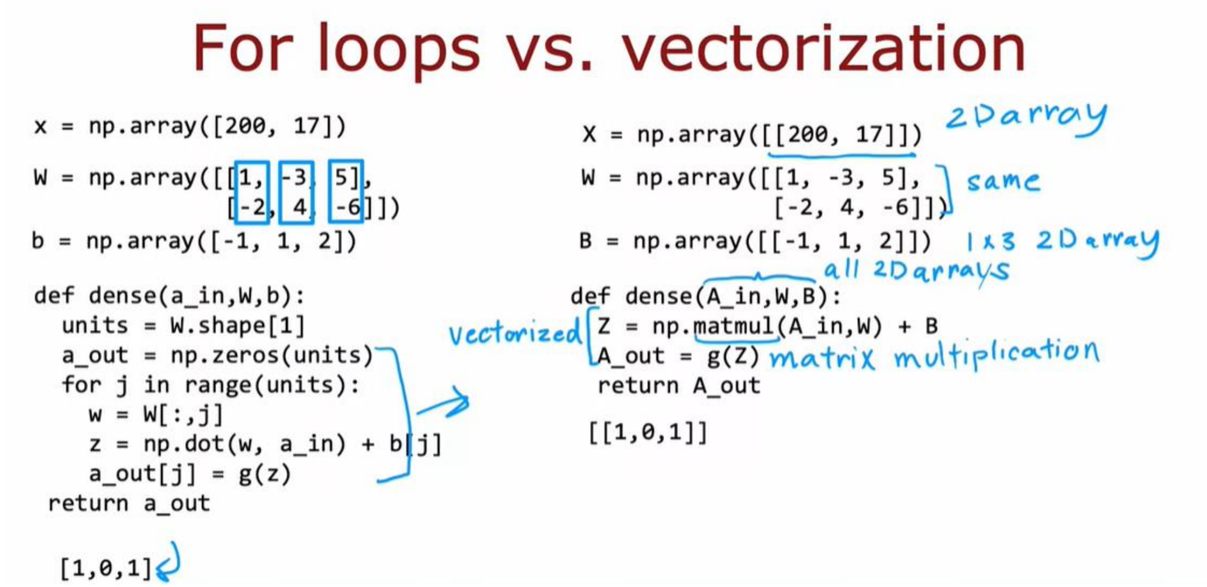




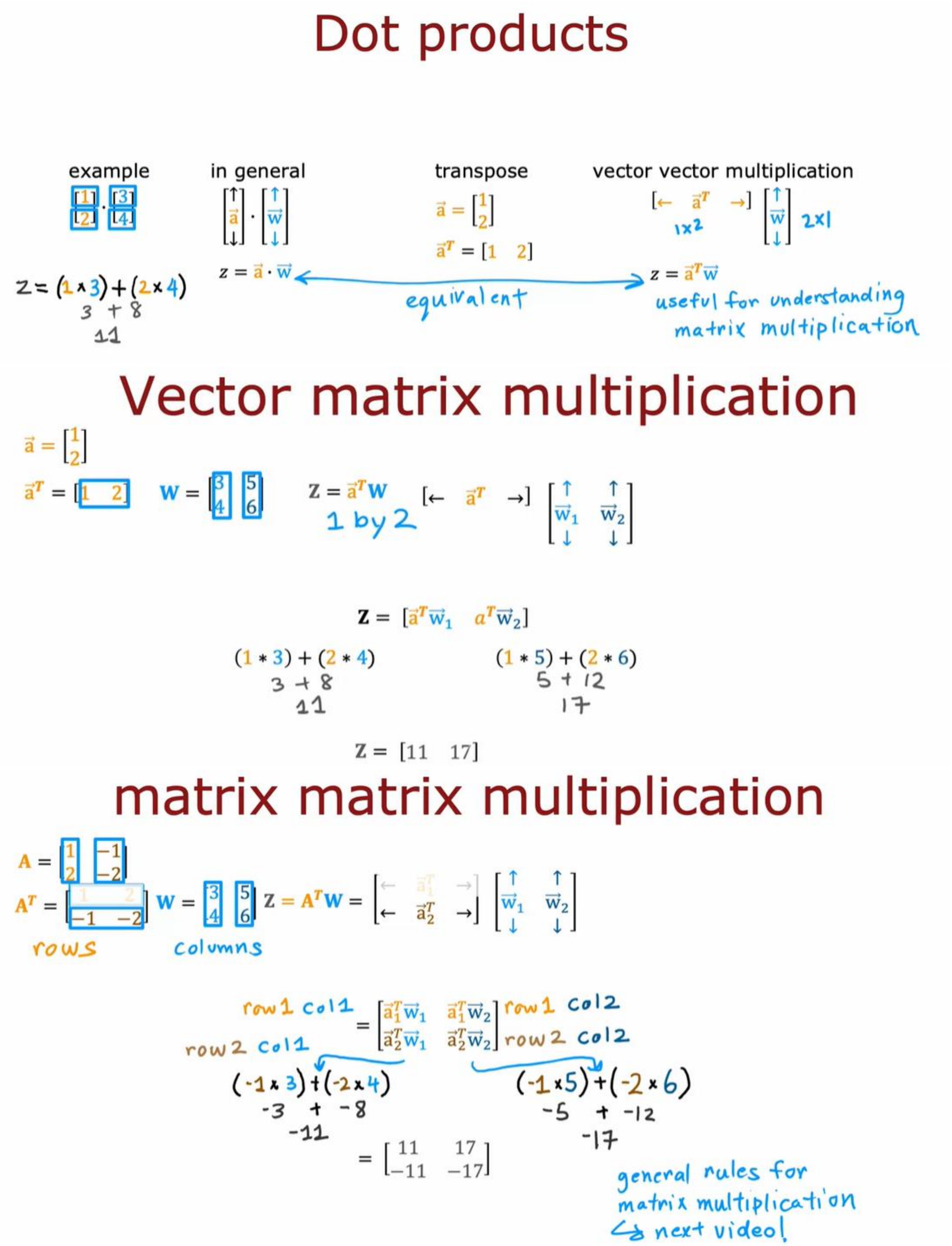


Vectorization

1. How neural network are implemented correctly



2. Matrix multiplication



3. Matrix multiplication rules

Matrix multiplication rules

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & \boxed{5} & 7 & 9 \\ 4 & \boxed{6} & 8 & 0 \end{bmatrix} \quad \mathbf{Z} = \mathbf{A}^T\mathbf{W} = \begin{bmatrix} 11 & 17 & 23 & 9 \\ -11 & -17 & -23 & -9 \\ 1.1 & 1.7 & 2.3 & 0.9 \end{bmatrix}$$

3 by 4 matrix

$\vec{a}_1^T \vec{w}_1 = (1 \times 3) + (2 \times 4) = 11$

row 3 column 2

$\vec{a}_3^T \vec{w}_2 = (0.1 \times 5) + (0.2 \times 6) = 1.7$

0.5 + 1.2

row 2 column 3?

$\vec{a}_2^T \vec{w}_3 = (-1 \times 7) + (-2 \times 8) = -23$

-7 + -16

Matrix multiplication rules

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & \boxed{5} & 7 & 9 \\ 4 & \boxed{6} & 8 & 0 \end{bmatrix} \quad \mathbf{Z} = \mathbf{A}^T\mathbf{W} = \begin{bmatrix} 11 & 17 & 23 & 9 \\ -11 & -17 & -23 & -9 \\ 1.1 & 1.7 & 2.3 & 0.9 \end{bmatrix}$$

3×2 2×4

can only take dot products of vectors that are same length

$[0.1 \;\; 0.2]$ length 2 $\begin{bmatrix} 5 \\ 6 \end{bmatrix}$ length 2

3 by 4 matrix

same #rows as A^T

same #columns as W

4. Matrix multiplication code

Matrix multiplication in NumPy

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0.1 \\ 2 & -2 & 0.2 \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 2 \\ -1 & -2 \\ 0.1 & 0.2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 0 \end{bmatrix} \quad \mathbf{Z} = \mathbf{A}^T\mathbf{W} = \begin{bmatrix} 11 & 17 & 23 & 9 \\ -11 & -17 & -23 & -9 \\ 1.1 & 1.7 & 2.3 & 0.9 \end{bmatrix}$$

```
A=np.array([[1,-1,0.1],
            [2,-2,0.2]])

AT=np.array([[1,2],
             [-1,-2],
             [0.1,0.2]])

AT=A.T
```

transpose

```
W=np.array([[3,5,7,9],
            [4,6,8,0]])
```

```
Z = np.matmul(AT,W)
```

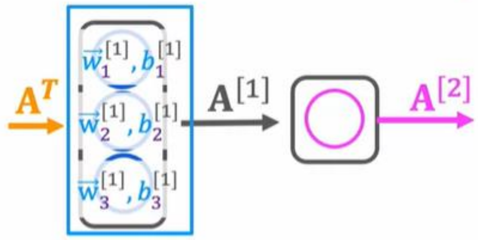
or

```
Z = AT @ W
```

result

```
[[11,17,23,9],
 [-11,-17,-23,-9],
 [1.1,1.7,2.3,0.9]
]
```

Dense layer vectorized



$A^T = [200 \quad 17]$

1×2

$$W = \begin{bmatrix} \boxed{\begin{matrix} 1 \\ -2 \end{matrix}} & \boxed{\begin{matrix} -3 \\ 4 \end{matrix}} & \boxed{\begin{matrix} 5 \\ -6 \end{matrix}} \end{bmatrix}$$

2×3

$B = [-1 \quad 1 \quad 2]$

1×3

$$Z = A^T W + B$$

$$\begin{bmatrix} \boxed{165} & \boxed{-531} & \boxed{900} \end{bmatrix}$$

$$\begin{matrix} z_1^{[1]} & z_2^{[1]} & z_3^{[1]} \end{matrix}$$

$$A = g(Z)$$

$$[1 \quad 0 \quad 1]$$

A

```
AT = np.array([[200, 17]])
W = np.array([[1, -3, 5],
              [-2, 4, -6]])
b = np.array([[-1, 1, 2]])
```

a_in

```
def dense(AT,W,b):
  z = np.matmul(AT,W) + b
  a_out = g(z)
  return a_out
```

a_in

```
[[1,0,1]]
```