Intro

Monday, 13 February, 2023 7:24 PM

→ In traditional development

Known: Input, Algorithm / formula

Unknown: Output.

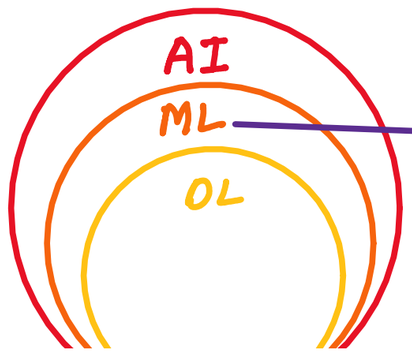
Example:

```
def (C) :
        f = C * (9/5) + 32   → RULE
        return f
```

→ In AI / ML / DL:

Known : Input, Output

Unknown : Algorithm / formula / pattern

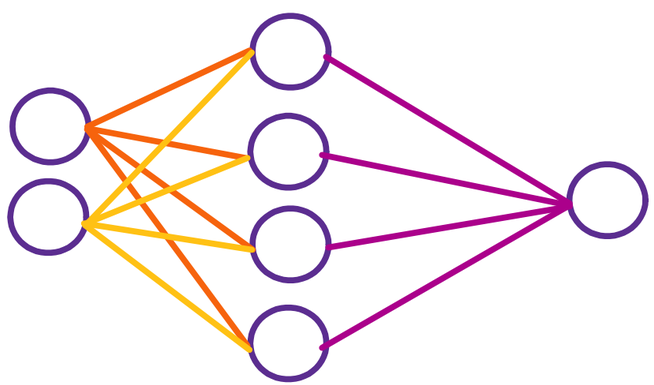


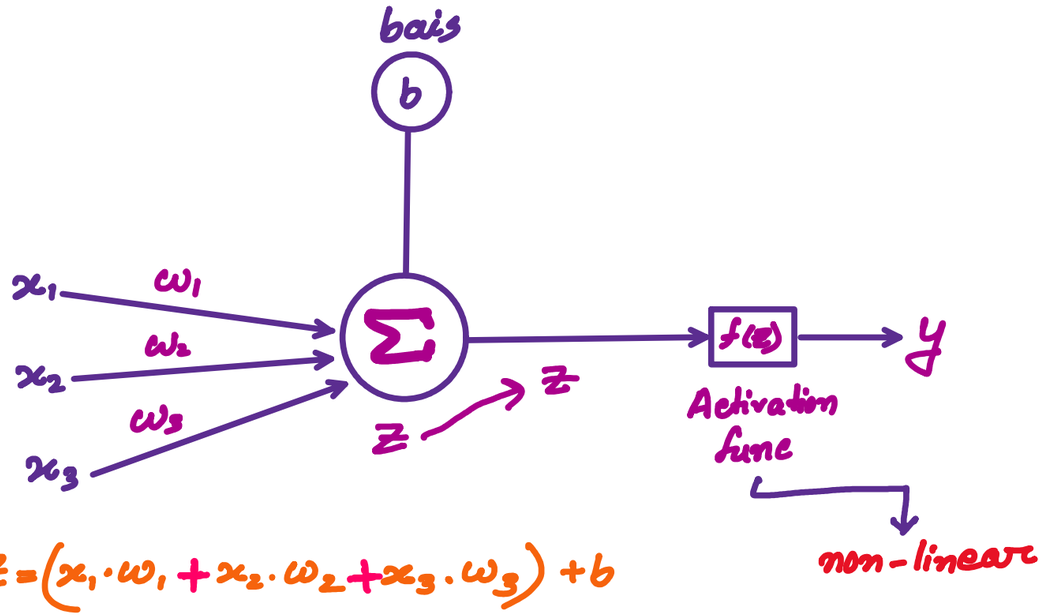
→ math heavy

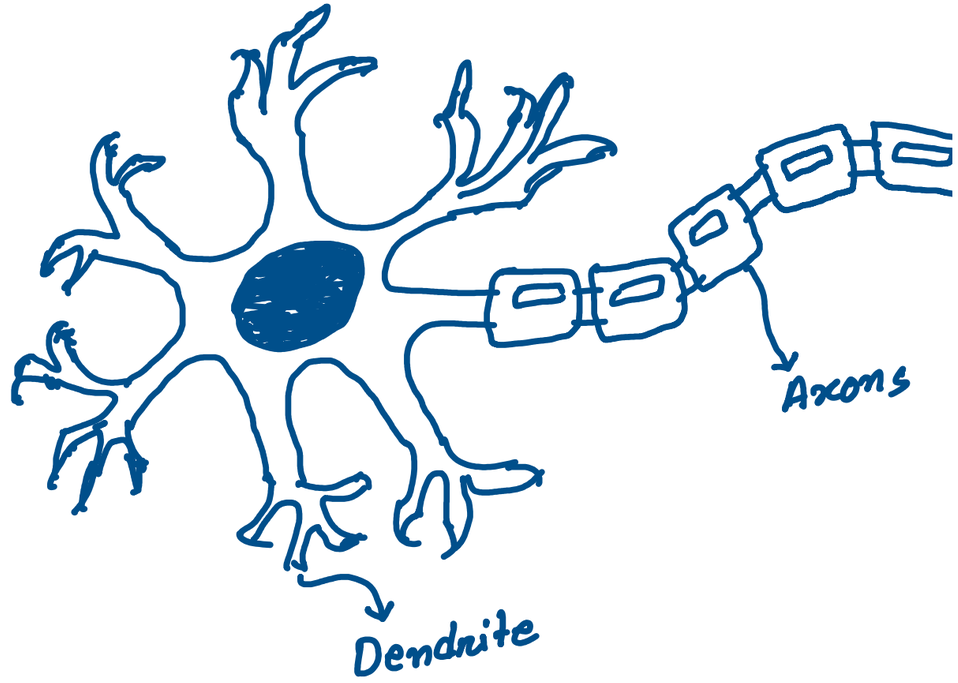
$$J(\theta) = -\frac{1}{m}\left[\sum_{k=1}^{k}\sum_{i=1}^{m} y_k^{(i)} \log\left(h_\theta(x^{(i)})\right)_k + \left(1-y_k^{(i)}\right) \log\left(1-\left(h_\theta(x^{(i)}\right.\right.\right.$$

$$\theta_j := \theta_j - \frac{\alpha}{m}\left[\sum^{m}\right.$$

→ Neural Network:







↳ $z = mx + c$

↳ Straight line

↳ Linear

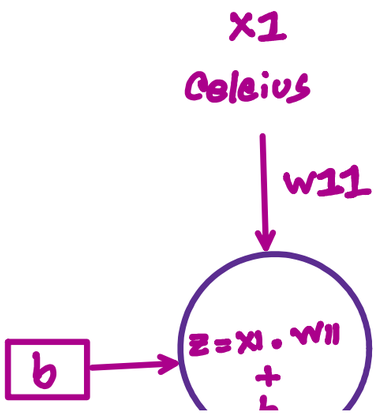
Lets Know Weights and Bias

Single layer:

num of neurons

```
l0 = tf.keras.layers.Dense(units=1,
                           input_shape=[1])
```

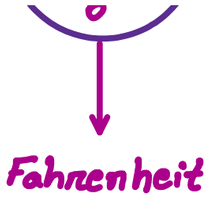
single value input



F = C * 1.8 + 32

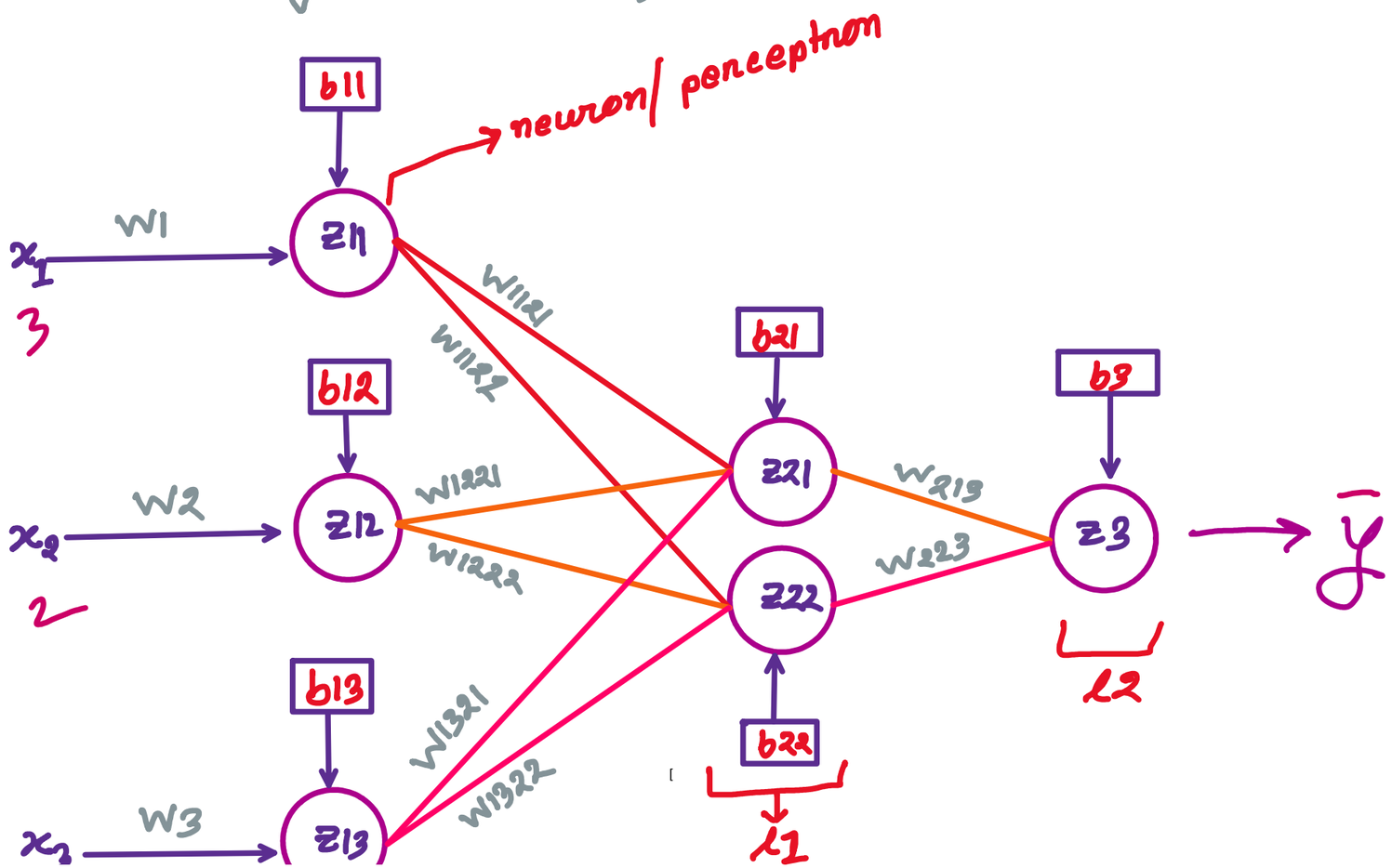
z = x1 * w11 + b

- Zero initialization → `w[l] = np.random.`
- Random initialization → `w[l] = np.random.`
- He initialization → `w[l] = np.random.`
  - ↳ $\sqrt{\frac{2}{size[l-1]}}$
- Xavier initialization → `w[l] = np.random`



3- layers :

```
l0 = tf.keras.layers.Dense(units=3, input_shape=[1])
l1 = tf.keras.layers.Dense(units=2)
l2 = tf.keras.layers.Dense(units=1)
```



$z_{11} = x_1 \cdot w_1 + b_{11}$; $z_{12} = x_2 \cdot w_2 + b_{12}$

$z_{13} = x_3 \cdot w_3 + b_{13}$

$z_{21} = z_{11} \cdot w_{1121} + z_{12} \cdot w_{1221} + z_{13} \cdot w_{1321} + b_{21}$

$z_{22} = z_{11} \cdot w_{1122} + z_{12} \cdot w_{1222} + z_{13} \cdot w_{1322} + b_{22}$

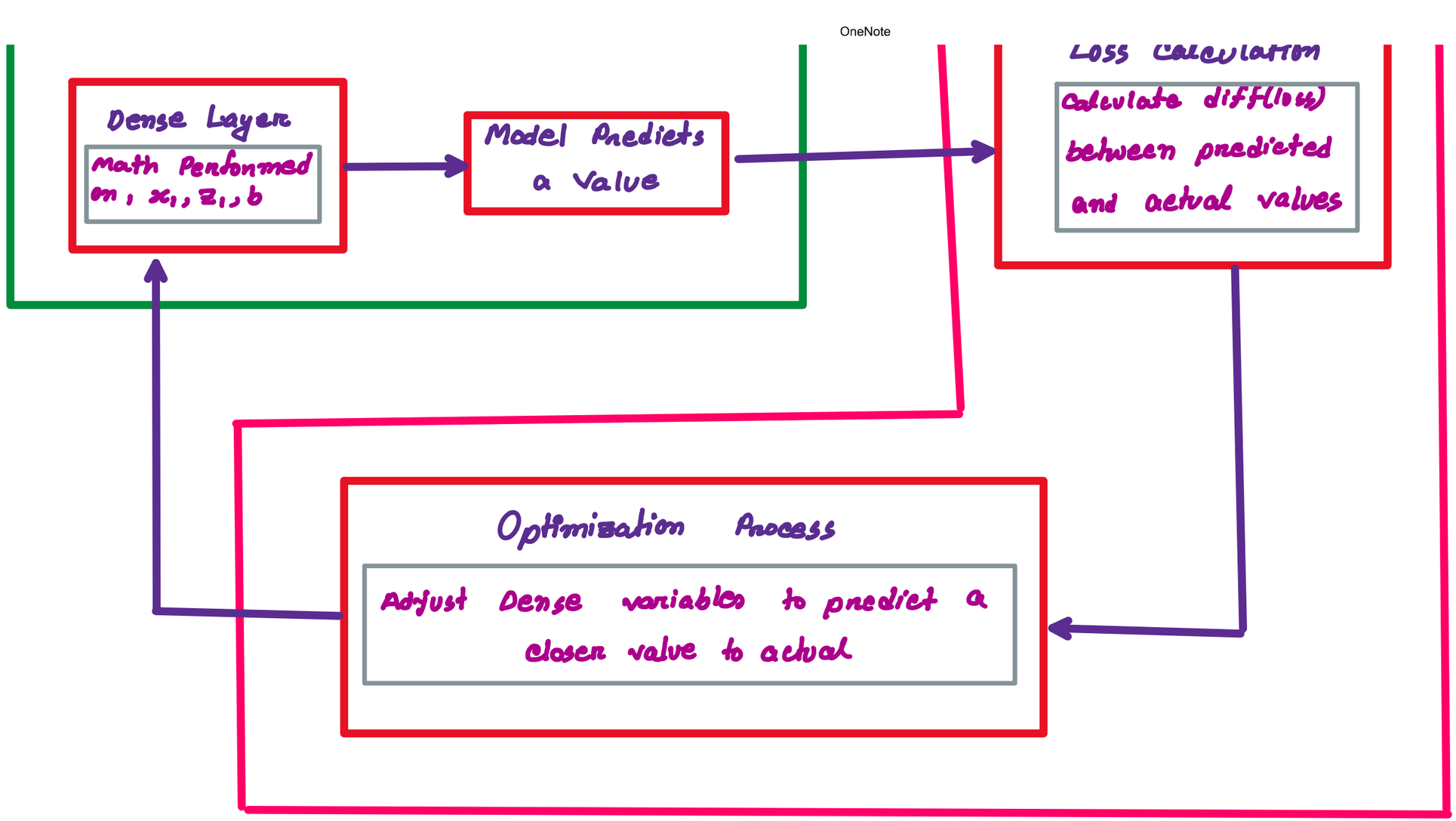
$z_3 = z_{21} \cdot w_{213} + z_{22} \cdot w_{223} + b_3$

$\downarrow$

y

Forward Pass

BackPropagation



monai, pytorch