

Deep Learning | EX02

Back Propagation & FF Coding

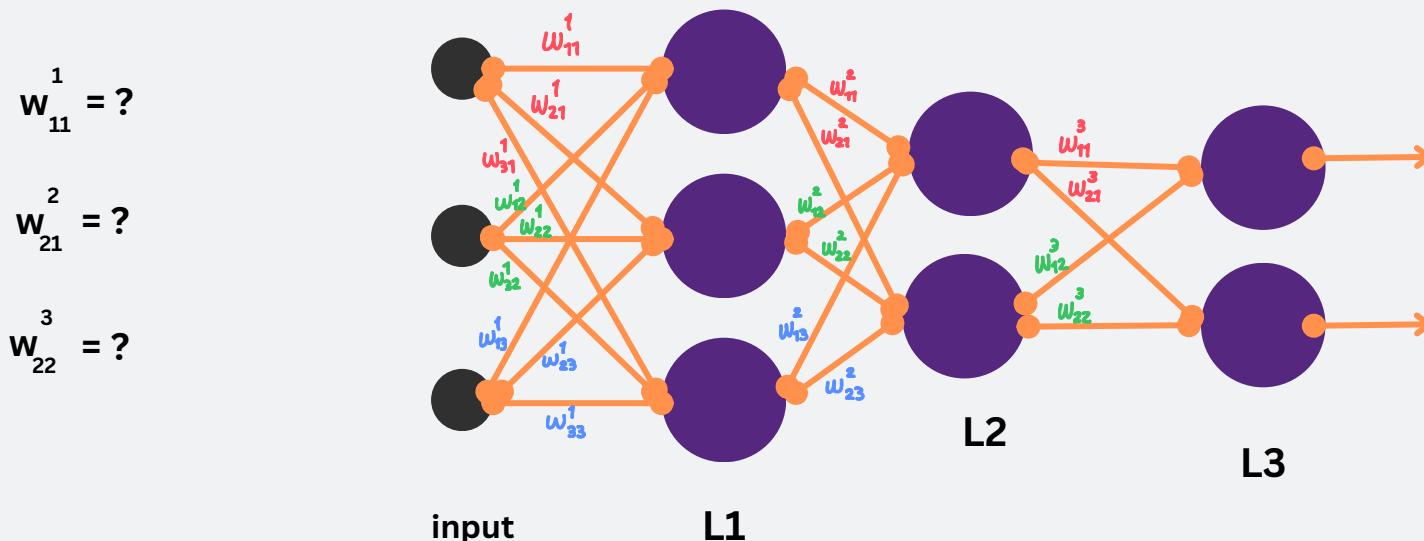
800

Computer Vision | Zahra Amini

Telegram: @zahraamini_ai & Instagram:@zahraamini_ai & LinkedIn: @zahraamini-ai

<https://zil.ink/zahraamini>

1. For the below feed-forward neural network architecture obtain the following weights after 1st Back Propagation iteration (epoch): [ignore the biases change during backpropagation]



inputs	L1-Weights	L1-Biases	L2-Weights	L2-Biases	L3-Weights	L3-Biases
$\begin{bmatrix} 0.3 \\ 0.5 \\ 0.8 \end{bmatrix}$	$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$	$\begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \end{bmatrix}$	$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$	$\begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$	$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$	$\begin{bmatrix} 0.5 \\ 0.3 \end{bmatrix}$

Activation Functions:

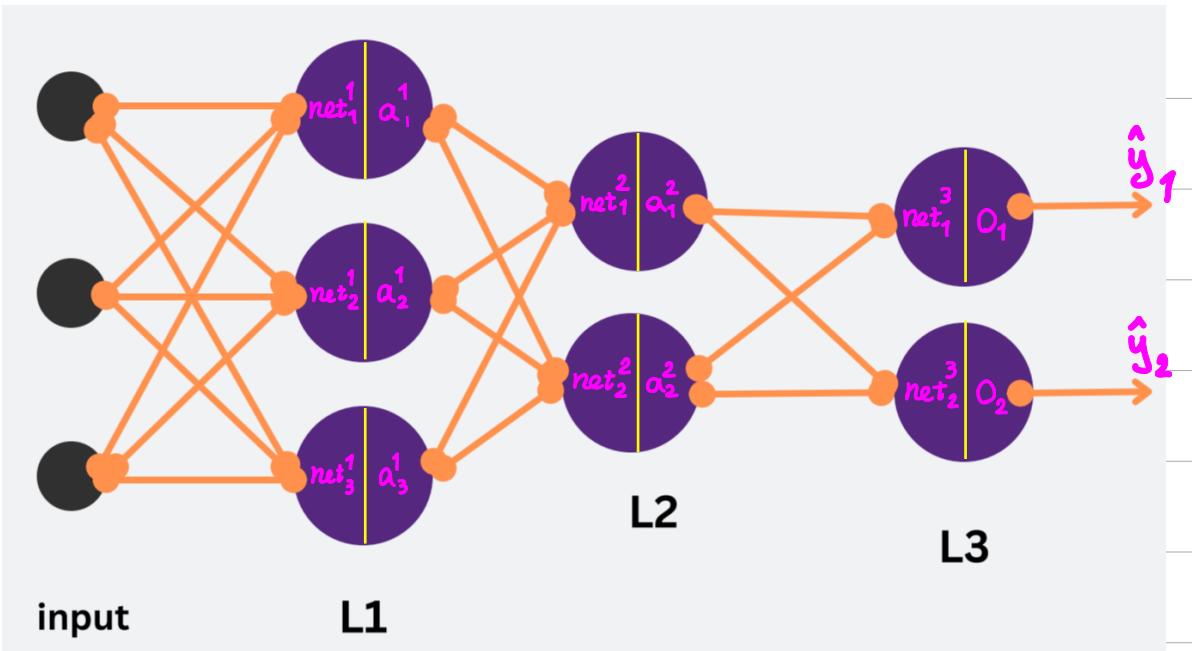
L1: Relu
L2: Tanh
L3: Sigmoid

Alpha=0.01

$$y_true = \begin{bmatrix} 0.5 \\ 0.8 \end{bmatrix}$$

Loss Function: MSE

Optimizer: GD



1. Feed Forward Pass:

Layer #1:

$$a_1^1 = \text{ReLU}(x_1 \cdot w_{11}^1 + x_2 \cdot w_{12}^1 + x_3 \cdot w_{13}^1 + b_1^1) = \text{ReLU}(0.3 \times 0.1 + 0.5 \times 0.2 + 0.8 \times 0.2 + 0.2) = \text{ReLU}(0.49) = 0.49$$

$$a_2^1 = \text{ReLU}(x_1 \cdot w_{21}^1 + x_2 \cdot w_{22}^1 + x_3 \cdot w_{23}^1 + b_2^1) = \text{ReLU}(0.3 \times 0.4 + 0.5 \times 0.1 + 0.8 \times 0.2 + 0.2) = \text{ReLU}(0.53) = 0.53$$

$$a_3^1 = \text{ReLU}(x_1 \cdot w_{31}^1 + x_2 \cdot w_{32}^1 + x_3 \cdot w_{33}^1 + b_3^1) = \text{ReLU}(0.3 \times 0.2 + 0.5 \times 0.3 + 0.8 \times 0.1 + 0.1) = \text{ReLU}(0.39) = 0.39$$

Layer #2:

$$a_1^2 = \tanh(a_1^1 \cdot W_{11}^2 + a_2^1 \cdot W_{12}^2 + a_3^1 \cdot W_{13}^2 + b_1^2) = \tanh(0.49 \times 0.1 + 0.53 \times 0.2 + 0.39 \times 0.2 + 0.1) = \tanh(0.333) \approx 0.321$$

$$a_2^2 = \tanh(a_1^1 \cdot W_{21}^2 + a_2^1 \cdot W_{22}^2 + a_3^1 \cdot W_{23}^2 + b_2^2) = \tanh(0.49 \times 0.4 + 0.53 \times 0.1 + 0.39 \times 0.2 + 0.4) = \tanh(0.727) \approx 0.621$$

Layer #3:

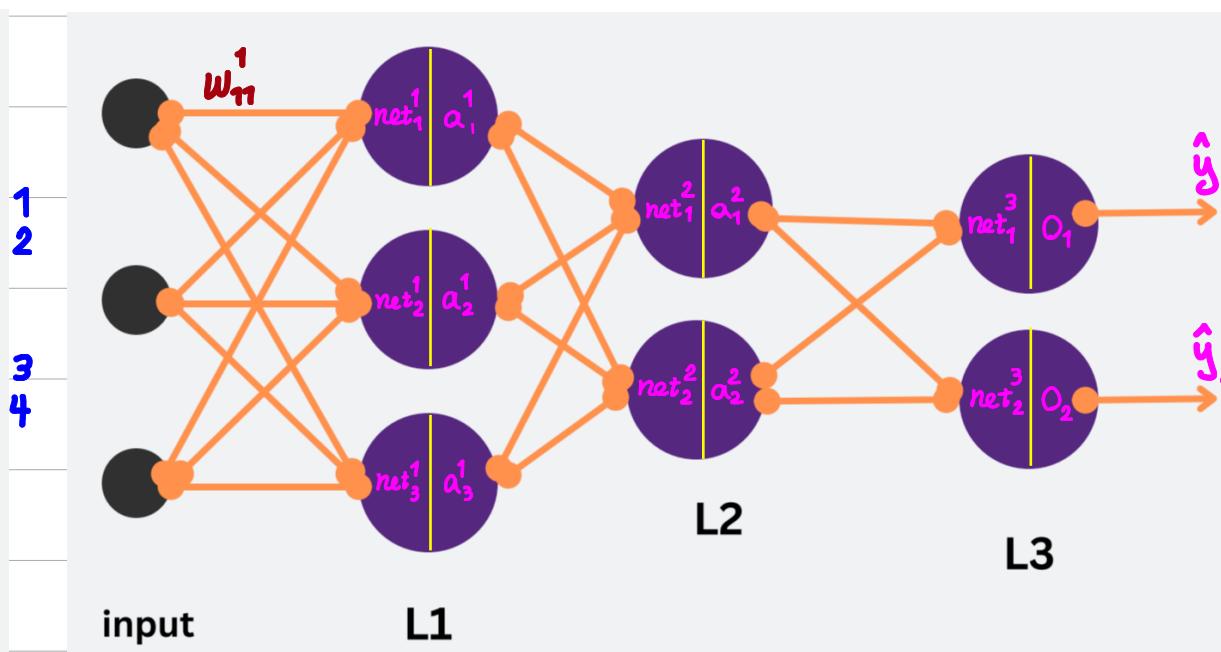
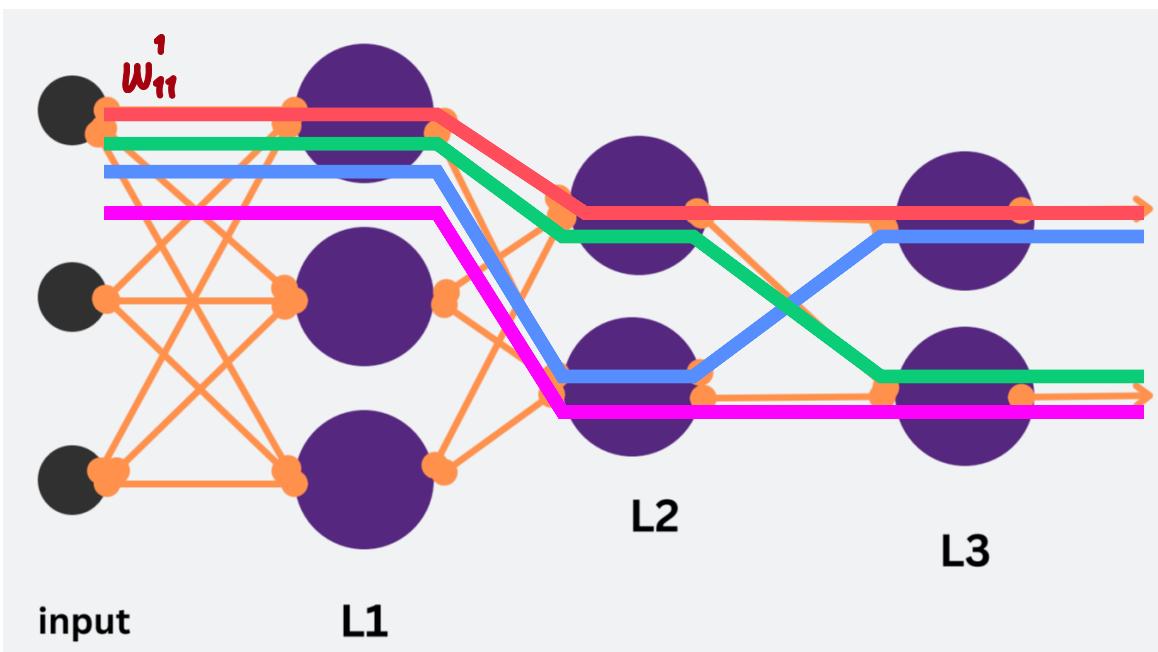
$$O_1 = \text{Sigmoid}(a_1^2 \cdot W_{11}^3 + a_2^2 \cdot W_{12}^3 + b_1^3) = \text{Sigmoid}(0.321 \times 0.5 + 0.621 \times 0.1 + 0.5) = \text{Sigmoid}(0.7226) = 0.673$$

$$O_2 = \text{Sigmoid}(a_1^2 \cdot W_{21}^3 + a_2^2 \cdot W_{22}^3 + b_2^3) = \text{Sigmoid}(0.321 \times 0.5 + 0.621 \times 0.5 + 0.3) = \text{Sigmoid}(0.771) = 0.684$$

2. Calculate Loss :

$$\text{Loss} = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 \rightarrow \text{Loss} = ((0.5 - 0.573)^2 + (0.8 - 0.628)^2) \div 2 \approx 0.0174$$

3. Backpropagation :



$$W_{11}^1 \text{ new} = W_{11}^1 \text{ old} - \alpha \frac{\delta L}{\delta w_{11}^1}$$

? $\frac{\delta L}{\delta w_{11}^1} = 1 + 2 + 3 + 4$

- ① $\hat{y}_1 \rightarrow O_1 \rightarrow \text{net}_1^3 \rightarrow a_1^2 \rightarrow \text{net}_1^2 \rightarrow a_1^1 \rightarrow \text{net}_1^1 \rightarrow w_{11}^1$
- ② $\hat{y}_1 \rightarrow O_1 \rightarrow \text{net}_2^3 \rightarrow a_2^2 \rightarrow \text{net}_2^2 \rightarrow a_2^1 \rightarrow \text{net}_1^1 \rightarrow w_{11}^1$
- ③ $\hat{y}_2 \rightarrow O_2 \rightarrow \text{net}_2^3 \rightarrow a_1^2 \rightarrow \text{net}_1^2 \rightarrow a_1^1 \rightarrow \text{net}_1^1 \rightarrow w_{11}^1$
- ④ $\hat{y}_2 \rightarrow O_2 \rightarrow \text{net}_2^3 \rightarrow a_2^2 \rightarrow \text{net}_2^2 \rightarrow a_2^1 \rightarrow \text{net}_1^1 \rightarrow w_{11}^1$

$$\frac{\delta L}{\delta W_{11}^1} = \left[\begin{array}{ccccccccc} \frac{\delta L}{\delta \hat{y}_1} & \frac{\delta \hat{y}_1}{\delta O_1} & \frac{\delta O_1}{\delta net_1^3} & \frac{\delta net_1^3}{\delta a_1^2} & \frac{\delta a_1^2}{\delta net_1^2} & \frac{\delta net_1^2}{\delta a_1^1} & \frac{\delta a_1^1}{\delta net_1^1} & \frac{\delta net_1^1}{\delta W_{11}^1} \end{array} \right] + \left[\begin{array}{ccccccccc} \frac{\delta L}{\delta \hat{y}_1} & \frac{\delta \hat{y}_1}{\delta O_1} & \frac{\delta O_1}{\delta net_2^3} & \frac{\delta net_2^3}{\delta a_2^2} & \frac{\delta a_2^2}{\delta net_2^2} & \frac{\delta net_2^2}{\delta a_2^1} & \frac{\delta a_2^1}{\delta net_2^1} & \frac{\delta net_2^1}{\delta W_{11}^1} \end{array} \right] + \\ \left[\begin{array}{ccccccccc} \frac{\delta L}{\delta \hat{y}_2} & \frac{\delta \hat{y}_2}{\delta O_2} & \frac{\delta O_2}{\delta net_2^3} & \frac{\delta net_2^3}{\delta a_1^2} & \frac{\delta a_1^2}{\delta net_1^2} & \frac{\delta net_1^2}{\delta a_1^1} & \frac{\delta a_1^1}{\delta net_1^1} & \frac{\delta net_1^1}{\delta W_{11}^1} \end{array} \right] + \left[\begin{array}{ccccccccc} \frac{\delta L}{\delta \hat{y}_2} & \frac{\delta \hat{y}_2}{\delta O_2} & \frac{\delta O_2}{\delta net_2^3} & \frac{\delta net_2^3}{\delta a_2^2} & \frac{\delta a_2^2}{\delta net_2^2} & \frac{\delta net_2^2}{\delta a_2^1} & \frac{\delta a_2^1}{\delta net_2^1} & \frac{\delta net_2^1}{\delta W_{11}^1} \end{array} \right]$$

$$\frac{\delta net_1^1}{\delta W_{11}^1} = \frac{\partial}{\partial W_{11}^1} \left(x_1 \cdot W_{11}^1 + \underset{\theta}{x_2} \cdot \cancel{W_{12}^1} + \cancel{x_3 \cdot W_{13}^1 + b_1} \right) = x_1 = 0.3 \quad \frac{a_1^1}{net_1^1} = \frac{\partial}{\partial net_1^1} \left(\text{ReLU}(net_1^1) \right) = 1^{0.49}$$

$$* \text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$\frac{\delta net_1^2}{\delta a_1^1} = \frac{\partial}{\partial a_1^1} \left(a_1^1 \cdot W_{11}^2 + \underset{\theta}{a_2^1} \cdot \cancel{W_{12}^2} + \cancel{a_3^1 \cdot W_{13}^2 + b_2} \right) = W_{11}^2 = 0.1$$

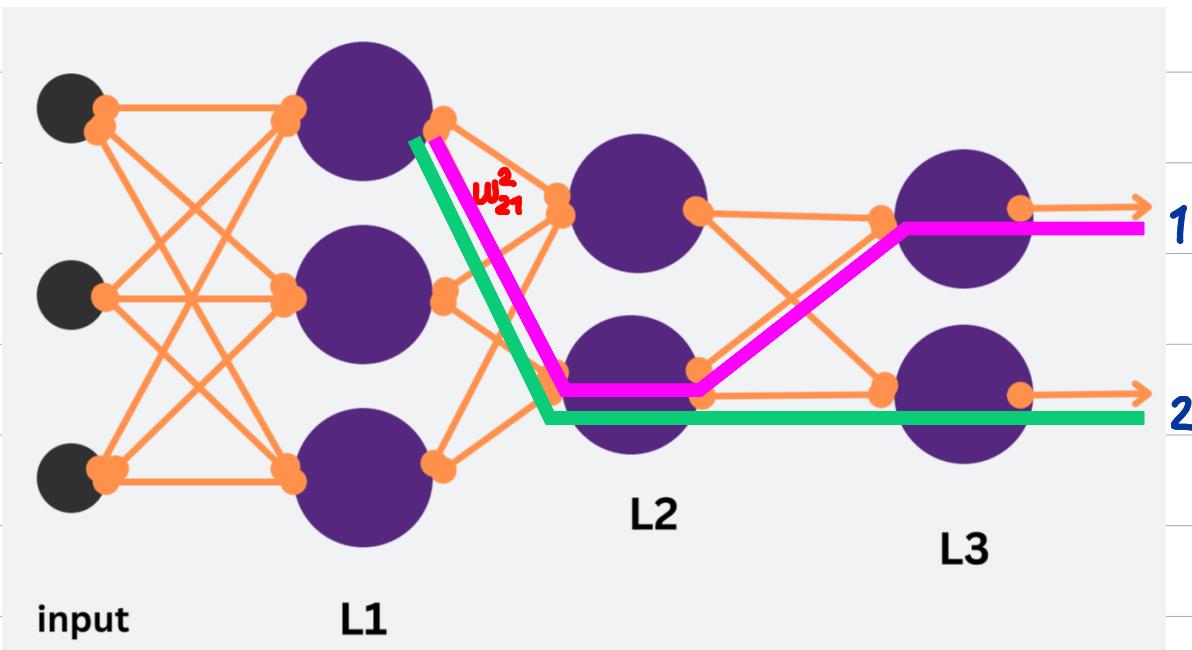
$$\frac{\delta a_1^2}{\delta net_1^2} = \frac{\partial}{\partial net_1^2} \left(\tanh(net_1^2) \right) = 1 - \tanh^2(\underset{\theta}{net_1^2}) = 0.897^{0.333}$$

$$\frac{\delta net_1^3}{\delta a_1^2} = \frac{\partial}{\partial a_1^2} \left(a_1^2 \cdot W_{11}^3 + \underset{\theta}{a_2^2} \cdot \cancel{W_{12}^3} + \cancel{b_3} \right) = W_{11}^3 = 0.5$$

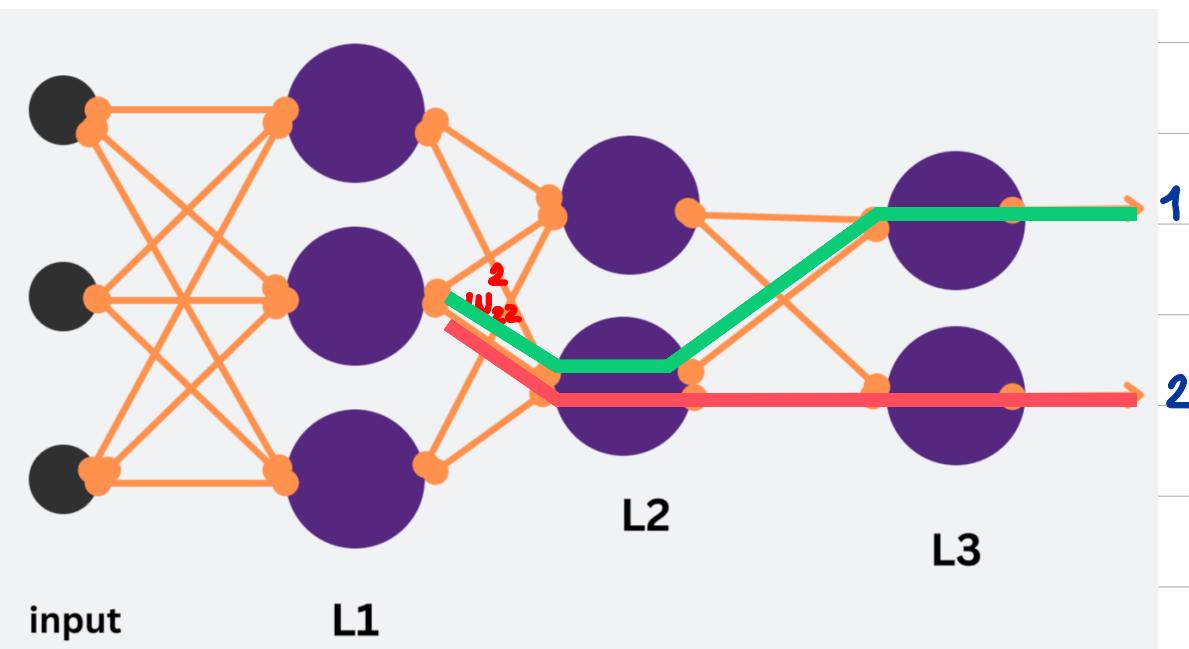
$$\frac{\delta O_1}{net_1^3} = \frac{\partial}{\partial net_1^3} (\text{Sigmoid}(net_1^3)) = \text{Sigmoid}(net_1^3) \cdot (1 - \text{Sigmoid}(net_1^3)) = 0.22$$

$$\frac{\delta \hat{y}_1}{\delta O_1} = 1 \quad \frac{\delta L}{\delta \hat{y}_1} = \frac{\partial}{\partial \hat{y}_1} \left(\frac{1}{m} (y_1 - \hat{y}_1)^2 \right) = \frac{2}{m} (y_1 - \hat{y}_1) (-1) = -\frac{2}{m} (y_1 - \hat{y}_1) = -2 (0.5 - 0.673) = 0.346$$

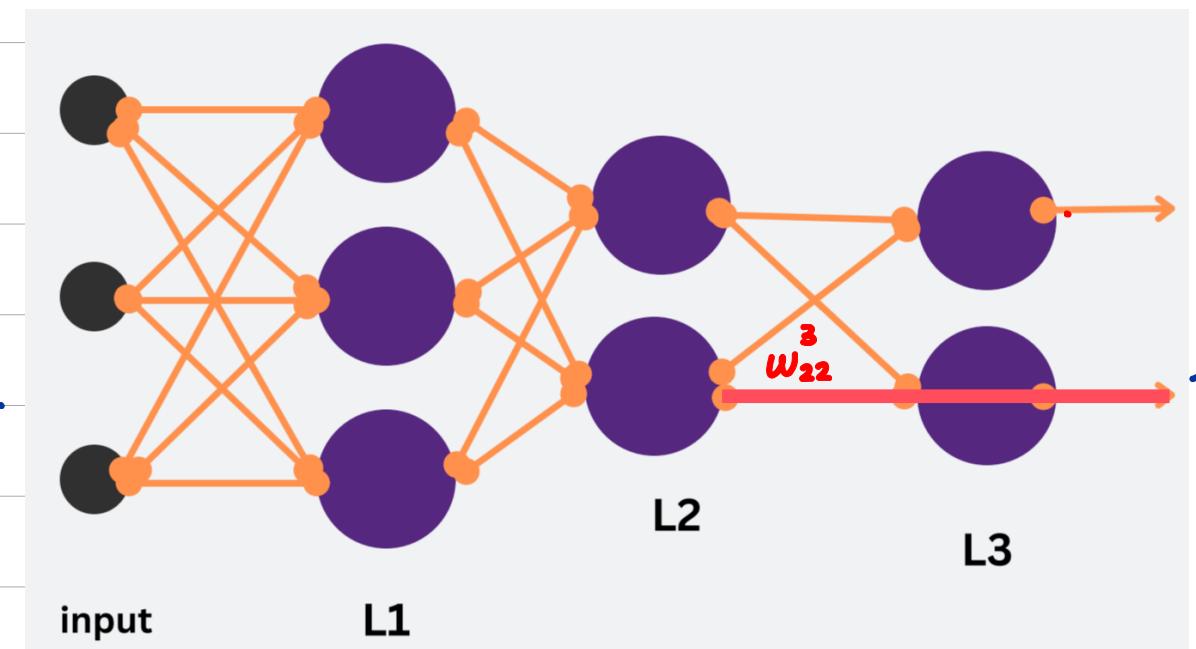
$$W_{21}^2$$



$$W_{22}^2$$



$$W_{22}^3$$



2. As you saw in the previous session, the designed model **could not predict the input image correctly**. Change the code and Design the **network using Dense layers** so that it can classify the input image (seven.png) correctly.

```
train_labels[0]
array([0., 0., 0., 0., 1., 0., 0., 0.], dtype=float32)

#Model
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

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

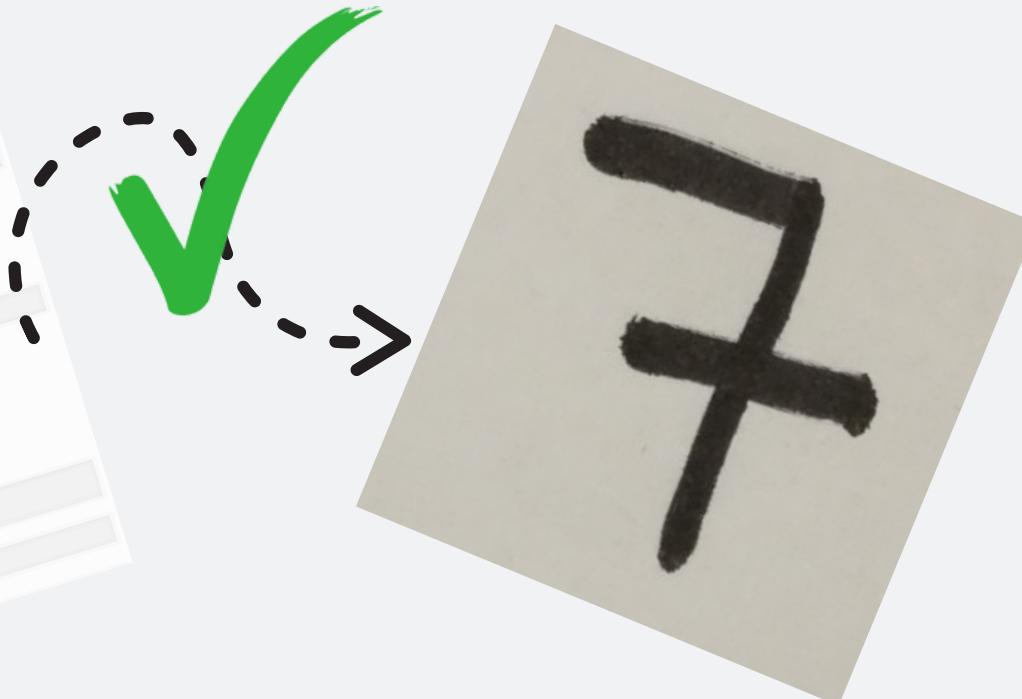
#Train
model.fit(train_images, train_labels, epochs=5, batch_size=32)

Epoch 1/5
1875/1875 [=====] - 5s 2ms/step - loss: 0.2411 - accuracy: 0.9292
Epoch 2/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.1825 - accuracy: 0.9695
Epoch 3/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.0737 - accuracy: 0.9776
Epoch 4/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.0530 - accuracy: 0.9833
Epoch 5/5
1875/1875 [=====] - 4s 2ms/step - loss: 0.0421 - accuracy: 0.9862


model.summary()

Model: "sequential_1"
Layer (type)          Output Shape       Param #
dense_3 (Dense)      (32, 128)         39840
dense_4 (Dense)      (32, 64)          8356
dense_5 (Dense)      (32, 10)          650
Total params: 109,396
Trainable params: 109,396
Non-trainable params: 0

#Evaluation
test_loss, test_acc = model.evaluate(test_images, test_labels)
313/313 [=====] - 1s 2ms/step - loss: 0.0056 - accuracy: 0.9752
test_loss
0.00564746379852295
```



Improving Model:

1. Increase Model Complexity

2. Hyperparameter Tuning

3. Increase Training Epochs

4. Data Augmentation

5. Model Regularization

- 1. Learning Rate
- 2. Batch Size
- 3. Optimizer
- 4. Number of Epochs

