HOMEWORK - 1

VIDUSHI BHATI
ID - 015309736

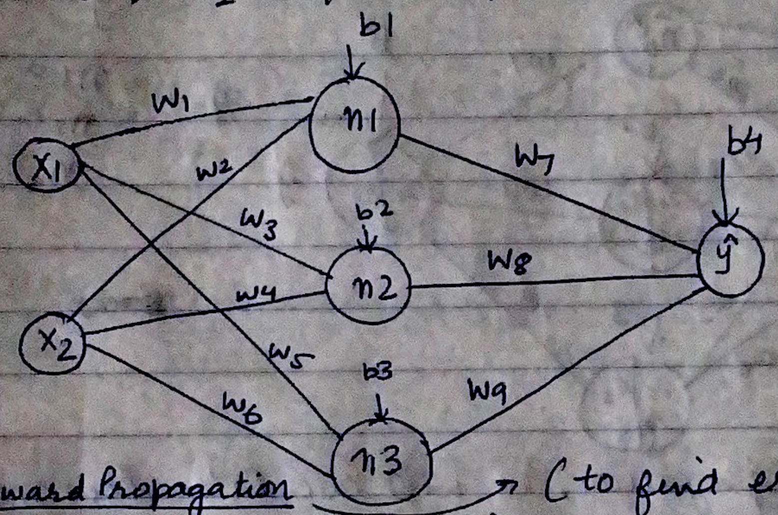
Q1 $\alpha = 0.1$ (learning rate)

σ = sigmoid function $\left(\frac{1}{1+e^{-x}}\right)$

$MSE = \frac{1}{2}(y-\hat{y})^2$

$y = 1$

$x = 6$, $W_1 \to W_9 = 1$ $b_1 \to b_4 = 1$



$x_1 = 6$

$x2 = 1$

Forward Propagation → (to find error)

$n1 = \sigma(W_1 x_1 + W_2 x_2 + b1)$

$= \sigma(1.6 + 1.1 + 1) = \sigma(6+1+1) = \sigma(8) = \frac{1}{1+e^{-8}}$

$= 0.9997$

$n2 = \sigma(W_3 x_1 + W_4 x_2 + b2)$

$= \sigma(1.6 + 1.1 + 1) = \sigma(8) = \frac{1}{1+e^{-8}} = 0.9997$

$n3 = \sigma(w_5 . x_1 + w_6 x_2 + b3)$

$= \sigma(1.6 + 1.1 + 1) = \sigma(8) = \frac{1}{1+e^{-8}} = 0.9997$

$\hat{y} = \sigma(n1.w7 + n2.w8 + n3.w9 + b4)$

$= \sigma(0.9997.1 + 0.9997.1 + 0.9997.1 + 1) = \sigma(3.9989)$

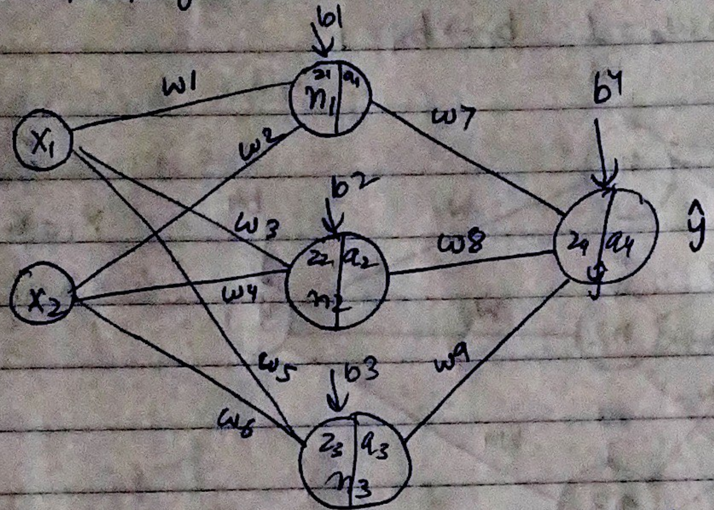
$= 0.98199$

Error $= y - \hat{y}$

$= 1 - 0.981996$

$= 0.01800 \approx 0.018$

Backpropagation (to adjust weights)



$y = 1$

Let:
$\hat{y} = a_4$
$n_1 = a_1$
$n_2 = a_2$
$n_3 = a_3$

For backpropagation, we will calculate how much of the Error is contributed by a particular weight and we will adjust the weight accordingly through gradient descent.

Error $= y - \hat{y}$ MSE $= \frac{1}{2}(y - \hat{y})^2$

In backpropagation, the parameters to be calculated are $w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9$
$b_1, b_2, b_3,$ and b_4.

We will use chain Rule to find each weight and bias.

1. $\frac{dE}{dW_7} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{dw_7}$ $\quad a_4 = \hat{y}$

$\frac{dE}{da_4} = \frac{d(\frac{1}{2}(y-\hat{y})^2)}{d\hat{y}} = \frac{\cancel{2}}{\cancel{2}} \cdot (y-\hat{y}) \cdot (-1) = -0.018$

$\frac{da_4}{dz_4} = \frac{d(\sigma_4(z_4))}{dz_4} = \sigma_4(1-\sigma_4) = a_4(1-a_4)$ $\quad \sigma_4 = a_4 = \hat{y}$

$= 0.981996\,(1-0.981996)$

$= \cancel{0.017679}\ 0.01768$

$\frac{dz_4}{dw_7} = \frac{d(n1.w7 + n2.w8 + n3.w9 + b4)}{dw7} = n1 = 0.9997$

$\frac{dE}{dw_7} = -0.018 \times 0.017679 \times 0.9997 = \cancel{-0.0031813}$ -3.181417×10^{-4}

$W_7' = w_7^{old} - \alpha \frac{dE}{dw_7^{old}} = 1 - 0.1(-0.0031813)$

$= 1.000031814$

2. $\frac{dE}{db_4} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{db_4}$

$= -0.018 \times 0.017679 \cdot \frac{dz_4}{db_4}$

$\frac{dz_4}{db_4} = \frac{d(n1.w7 + n2.w8 + n3.w9 + b4)}{db4} = 1$

$\frac{dE}{db_4} = -0.0031822 = -3.1824 \times 10^{-4}$

$b_4' = b_4^{old} - \alpha \frac{dz_4}{db_4^{old}} = 1 - 0.1(-0.0031822)$

$= 1.000031822$

3. $\frac{dE}{dw_8} = \frac{dE}{\cancel{d\hat{y}}\ da_4} \cdot \frac{\cancel{d\hat{y}}\ da_4}{dw_8} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{dw_8}$

$$\frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{dw_8} \qquad a_4 = \hat{y}$$

$$\frac{dE}{da_4} = \frac{d\left(\frac{1}{2}(y-\bar{y})^2\right)}{d\bar{y}} = \frac{2}{2}(y-\bar{y}) \cdot (-1)$$

$$= 1 - 0.981996 = -0.018$$

$$\frac{da_4}{dz_4} = \frac{d(\sigma(z_4))}{dz_4} = \sigma(1-\sigma) = 0.981996(1-0.981996)$$

$$= 0.017679$$

$$\frac{dz_4}{dw_8} = \frac{d(n1.w7 + n2.w8 + n3.w9 + b4)}{dw_8} = n2$$

$$\frac{dz_4}{dw_8} = 0.9997$$

$$\frac{dE}{dw_8} = -0.018 \times 0.017679 \times 0.9997 = -0.00031814$$

$$W_8' = W_8^{old} - \alpha \frac{dE}{dw_8} = 1 - 0.1 \times (-0.00031813)$$

$$= 1.00031814$$

(4) $$\frac{dE}{dw_9} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{dw_9}$$

$$\frac{dE}{da_4} = -0.018 \qquad \frac{da_4}{dz_4} = 0.17679$$

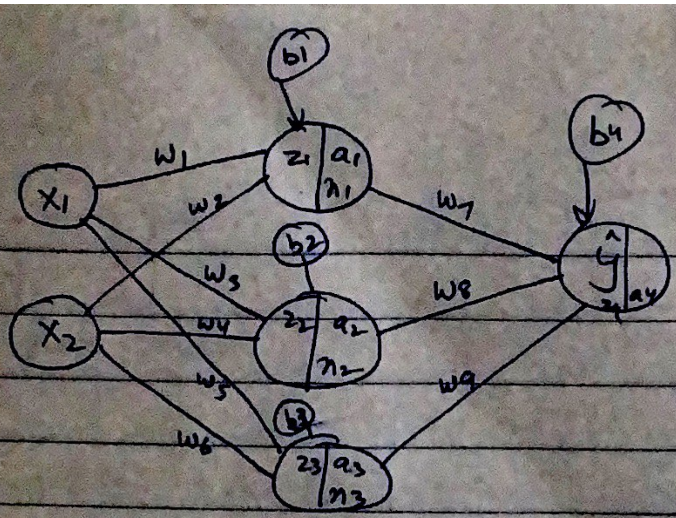
$$\frac{dz_4}{dw_9} = \frac{d(n1.w7 + n2.w8 + n3.w9 + b4)}{dw_9} = n3$$

$$= 0.9997$$

$$\therefore \frac{dE}{dw_9} = 0.17679 \times -0.018 \times 0.9997 = -0.0031814$$

$$W_9' = W_9^{old} - \alpha \frac{dE}{dw_9} = 1 - (0.1)(-0.0031814)$$

$$= w_9 = 1.00031814$$



\# $\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$

$\frac{\partial E}{\partial a_4} = -0.018 \qquad \frac{\partial a_4}{\partial z_4} = 0.017679$

$a_1 = n_1$
$a_2 = n_2$
$a_3 = n_3$

$\frac{\partial z_4}{\partial a_1} = \frac{\partial(w_7 a_1 + w_8 . a_2 + w_9 . a_3 + b_4)}{\partial a_1} = w_7 = 1$

$\frac{\partial a_1}{\partial z_1} = \frac{\partial(\sigma_1(z_1))}{\partial z_1} = \sigma_1(1-\sigma_1) = a_1(1-a_1) = n_1(1-n_1)$

$= 0.9997(1-0.9997)$

$= 2.999 \times 10^{-4}$

$\frac{\partial z_1}{\partial w_1} = \frac{\partial(x_1 w_1 + x_2 w_2 + b_1)}{\partial w_1} = x_1 = 6$

$\frac{\partial E}{\partial w_1} = -0.018 \times 0.017679 \times 1 \times 2.999 \times 10^{-4} \times 6$

$= -5.7261 \times 10^{-7}$

$w_1^{new} = w_1^{old} - 0.1(-5.7261 \times 10^{-7})$

$= 1 - 0.1(-5.7261 \times 10^{-7}) = 1.000000057$

\# $\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1}$

$\frac{\partial z_1}{\partial b_1} = \frac{\partial(x_1 w_1 + \cancel{x_2 w_2} + b_1)}{\partial b_1} = 1$

$\frac{\partial E}{\partial b_1} = -0.018 \times 0.017679 \times 1 \times 2.999 \times 10^{-4} \times 1 = -9.5435 \times 10^{-8}$

$b_1^{new} = b_1^{old} - 0.1(-9.5435 \times 10^{-8}) = 1.00000001$

#
$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_2}$$

$$\frac{\partial z_1}{\partial w_2} = \frac{\partial(x_1 w_1 + x_2 w_2 + b_1)}{\partial w_2} = x_2 = 1$$

$$\frac{\partial E}{\partial w_2} = -0.018 \times 0.017679 \times 1 \times 2.99 \times 10^{-4} \times 1 = -9.5435 \times 10^{-8}$$

$$w_2^{new} = w_2^{old} - (0.1)(-9.5435 \times 10^{-8}) = 1.00000001$$

#
$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_3}$$

$$\frac{\partial z_4}{\partial a_2} = \frac{\partial(a_1 w_7 + a_2 w_8 + a_3 w_9 + b_4)}{\partial a_2} = w_8 = 1$$

$n_1 = a_1$
$n_2 = a_2$
$n_3 = a_3$

$$\frac{\partial a_2}{\partial z_2} = \frac{\partial(\sigma_2(z_2))}{\partial z_2} = \sigma_2(1-\sigma_2) = a_2(1-a_2) = n_2(1-n_2)$$
$$= 0.9997(1-0.9997)$$
$$= 2.999 \times 10^{-4}$$

$$\frac{\partial z_2}{\partial w_3} = \frac{\partial(x_1 w_3 + x_2 w_4 + b_2)}{\partial w_3} = x_1 = 6$$

$$\frac{\partial E}{\partial w_3} = -0.018 \times 0.017679 \times 1 \times 2.999 \times 10^{-4} \times 6$$
$$= -5.7261 \times 10^{-7}$$

$$w_3^{new} = w_3^{old} - (0.1)(-5.7261 \times 10^{-7}) = 1 - 0.1(-5.7261 \times 10^{-7})$$
$$= 1.000000057$$

#
$$\frac{\partial E}{\partial w_4} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_4}$$
$$= -0.018 \times 0.017679 \times 1 \times 2.999 \times 10^{-4} \times \frac{\partial z_2}{\partial w_4}$$

$$\frac{\partial z_2}{\partial w_4} = \frac{\partial(x_1 w_3 + x_2 w_4 + b_2)}{\partial w_4} = x_2 = 1$$

$$\frac{\partial E}{\partial w_4} = -9.5345 \times 10^{-8} \qquad w_4^{new} = w_4^{old} - 0.1(-9.5345 \times 10^{-8})$$
$$= 1.00000001$$

$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2}$

$$\frac{\partial z_2}{\partial b_2} = \frac{\partial(x_1 w_3 + x_2 w_4 + b_2)}{\partial b_2} = 1$$

$$\frac{\partial E}{\partial b_2} = -0.018 \times 0.17679 \times 1 \times 2.999 \times 10^{-4} \times 1$$
$$= -9.5435 \times 10^{-8}$$

$$b_2^{new} = b_2^{old} - (0.1)\left(\frac{\partial E}{\partial b_2}\right) = 1 - (0.1)(-9.5435 \times 10^{-8})$$
$$= 1.00000001$$

$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_5}$

$$\frac{\partial z_4}{\partial a_3} = \frac{\partial(w_7 a_1 + w_8 a_2 + w_9 a_3)}{\partial a_3} = w_9 = 1$$

$$\frac{\partial a_3}{\partial z_3} = \frac{\partial(\sigma_3(1-\sigma_3))}{\partial z_3} = \frac{\partial(a_3(1-a_3))}{\partial z_3} = 0.9997(1-0.9997) \quad a_3 = \sigma_3$$
$$= 2.999 \times 10^{-4}$$

$$\frac{\partial z_3}{\partial w_5} = \frac{\partial(w_5 x_1 + w_6 x_2 + b_3)}{\partial w_5} = x_1 = 6$$

$$\frac{\partial E}{\partial w_5} = -0.018 \times 0.017679 \times 1 \times 2.999 \times 10^{-4} \times 6$$
$$= -5.7261 \times 10^{-7}$$

$$w_5^{new} = w_5^{old} - 0.1(-5.7261 \times 10^{-7}) = 1 - 0.1(-5.7261 \times 10^{-7})$$
$$= 1.000000057$$

$\frac{\partial E}{\partial w_6} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_6}$

$$\frac{\partial z_3}{\partial w_6} = \frac{\partial(w_5 x_1 + w_6 x_2 + b_3)}{\partial w_6} = x_2 = 1$$

$$\frac{\partial E}{\partial w_6} = -0.018 \times 0.17679 \times 1 \times 2.999 \times 10^{-4} \times 1 = -9.5435 \times 10^{-8}$$

$$w_6^{new} = w_6^{old} - \alpha \frac{\partial E}{\partial w_6^{old}}$$

$$= 1 - (0.1)(-9.5435 \times 10^{-8}) = 1.00000001$$

$$\# \frac{\partial E}{\partial b^3} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial b_3}$$

$$\frac{\partial z_3}{\partial b_3} = \frac{\partial (w_5 x_1 + w_6 x_2 + b_3)}{\partial b_3} = 1$$

$$\frac{\partial E}{\partial b_3} = -0.018 \times 0.17679 \times 1 \times 2.999 \times 10^{-4} \times 1 = -9.5435 \times 10^{-8}$$

$$b_3^{new} = b_3^{old} - \alpha \frac{dE}{db_3^{old}} = 1 - (0.1)(-9.5435 \times 10^{-8})$$

$$= 1.00000001$$

Input $x_1 = 6$ $x_2 = 1$ $\alpha = 0.1$

OLD VALUES - PARAMETER	NEW VALUES - PARAMETER
$w_1 = 1$	$w_1^{new} = \cancel{1.000031814}$ 1.000000057
$w_2 = 1$	$w_2^{new} = 1.00000001$
$w_3 = 1$	$w_3^{new} = 1.000000057$
$w_4 = 1$	$w_4^{new} = 1.00000001$
$w_5 = 1$	$w_5^{new} = 1.000000057$
$w_6 = 1$	$w_6^{new} = 1.00000001$
$w_7 = 1$	$w_7^{new} = 1.000031814$
$w_8 = 1$	$w_8^{new} = 1.000031814$
$w_9 = 1$	$w_9^{new} = 1.000031814$
$b_1 = 1$	$b_1^{new} = 1.00000001$
$b_2 = 1$	$b_2^{new} = 1.00000001$
$b_3 = 1$	$b_3^{new} = 1.00000001$
$b_4 = 1$	$b_4^{new} = 1.000031814$

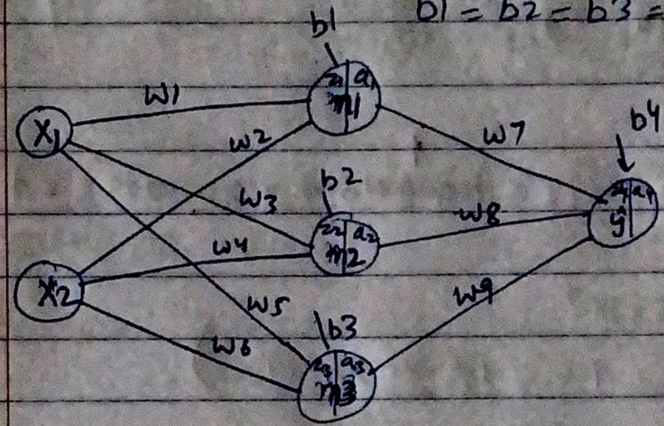
If all the values are initialized to be same, then there is hardly any change in updation, leading to extremely negligible convergence.

Q1. b) $x1 = 0$, $x2 = 6$, $w1 = w3 = w5 = 0.5$ $\alpha = 0.1$

$w2 = w4 = w6 = -0.5$ $y = 1$

$w7 = 1$ $w8 = -1$ $w9 = 0$ $MSE = \frac{1}{2}(y - \hat{y})$

$b1 = b2 = b3 = b4 = 0.1$



<u>Forward propagation</u>

$n1 = \sigma(z1) = \sigma(w1\,x1 + w2\,x_2 + b1)$

$= \sigma(0.5 \cdot 0 + (-0.5) \times 6 + 0.1)$

$= \sigma(0 - 3 + 0.1) = \sigma(-2.9)$

$= 0.0522 = a_1$

$n2 = \sigma(z2) = \sigma(x1 \cdot w3 + x2 \cdot w4 + b2)$

$= \sigma(0 + 6 \cdot (-0.5) + 0.1)$

$= \sigma(-3 + 0.1) = \sigma(-2.9) = 0.0522 = a_2$

$n3 = \sigma(z_3) = \sigma(x_1 \cdot w5 + x_2\,w6 + b3)$

$= \sigma(0 + 6 \cdot (-0.5) + 0.1)$

$= \sigma(-3 + 0.1) = \sigma(-2.9) = 0.0522 = a_3$

$n4 = \sigma(z_4) = \sigma(n1 \cdot w7 + n2 \cdot w8 + n3 \cdot w9 + b4)$

$= \sigma(0.0522 \cdot 1 + 0.0522 \cdot (-1) + 0.0522 \cdot 0 + 0.1)$

$\sigma(0.0\cancel{5}22 - 0.0\cancel{5}22 + 0 + 0.1)$

$= \sigma(0.1) = \frac{1}{1+e^{-0.1}} = 0.52498 = a_4 = \hat{y}$

$\hat{y} = 0.52498$

$y = 1$ $\quad E = y - \hat{y} = 0.4750$

$MSE = \frac{1}{2}(y-\hat{y})^2 \Rightarrow \frac{1}{2}(y-\hat{y})^2 =$ loss function.

Backpropagation

Calculate the updated weights and biases.

$\# \frac{dE}{dw_7} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{dw_7}$ $\quad a_4 = \hat{y}$

$\frac{dE}{da_4} = \frac{d\left(\frac{1}{2}(y-\hat{y})^2\right)}{d\hat{y}} = \frac{\cancel{2}}{\cancel{2}}(y-\hat{y})\cdot(-1) = \hat{y} - y = -0.4750$

$\frac{da_4}{dz_4} = \frac{d\sigma(z_4)}{dz_4} = \sigma(1-\sigma)$ $\quad \sigma = a_4 = n4$

$= 0.52498(1-0.52498)$

$= 0.2494$

$\frac{dz_4}{dw_7} = \frac{d(n1 \cdot w7 + n2\, w8 + n3 \cdot w9 + b4)}{dw_7} = n1 = 0.0522$

$\therefore \frac{dE}{dw_7} = -0.4750 \times 0.2494 \times 0.0522 = -6.184 \times 10^{-3}$

$w_7^{new} = w_7^{old} - \alpha \frac{dE}{dw_7} = 1 - 0.1 \times (-6.184 \times 10^{-3})$

$= 1.00062$

$\# \quad \frac{dE}{dw_8} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{dw_8}$

$\frac{dz_4}{dw_8} = \frac{d(x1.w7 + x2.w8 + x3.w9 + b4)}{dw_8} = x2 = a_2$

$= \underline{0.0522}$

$\therefore \frac{dE}{dw_8} = -0.4750 \times 0.2494 \times 0.0522 = -6.184 \times 10^{-3}$

$w_8^{new} = w_8^{old} - \alpha \frac{dE}{dw_8}$

$= -1 - 0.1(-6.184 \times 10^{-3})$

$= \underline{-0.9994}$

$\# \quad \frac{dE}{dw_9} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{dw_9}$

$\frac{dz_4}{dw_9} = \frac{d(x1.w7 + x2.w8 + x3.w9 + b4)}{dw_9} = x3 = a_3$

$= 0.0522$

$\frac{dE}{dw_9} = -0.4750 \times 0.2494 \times 0.0522 = -6.184 \times 10^{-3}$

$w_9^{new} = w_9^{old} - \alpha \frac{dE}{dw_9}$

$w_9^{new} = 0 - 0.1(-6.184 \times 10^{-3}) = \underline{6.184 \times 10^{-4}}$

$\# \quad \frac{dE}{db_4} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{db_4}$

$\frac{dz_4}{db_4} = \frac{d(x1.w7 + x2.w8 + x3.w9 + b4)}{db_4} = 1$

$$\frac{dE}{db_4} = -0.4750 \times 0.2494 \times 1 = -0.1185$$

$$b_4^{new} = b_4^{old} - \alpha \frac{dE}{db_4}$$

$$= 0.1 - (0.1)(-0.1185)$$

$$b_4^{new} = \underline{0.1118}$$

$\frac{dE}{dw_1} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{da_1} \cdot \frac{da_1}{dz_1} \cdot \frac{dz_1}{dw_1}$

$$\frac{dz_4}{da_1} = \frac{d(n1 \cdot w7 + n2 \cdot w8 + n3 w9 + b4)}{da_1} = w7 \qquad a_1 = n_1$$

$$= w_7 = 1$$

$$\frac{da_1}{dz_1} = \frac{d(\sigma(z1))}{dz_1} = \sigma(1-\sigma) = \qquad \sigma = a_1 = n_1$$

$$= 0.0522(1 - 0.0522)$$

$$= 0.04948$$

$$\frac{dz_1}{dw_1} = \frac{d(x_1 \cdot w_1 + x_2 w2 + b1)}{dw_1} = x_1 = 0$$

$$\frac{dE}{dw_1} = -0.4750 \times 0.2494 \times 1 \times 0.04948 \times 0 = 0$$

$$w_1^{new} = w_1^{old} - \alpha \frac{dE}{dw_1} = 0.5 - (0.1) \times 0$$

$$= 0.5$$

$$w_1^{new} = 0.5$$

$\frac{dE}{w_2} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{da_1} \cdot \frac{da_1}{dz_1} \cdot \frac{dz_1}{dw_2}$

$$\frac{dz_1}{dw_2} = \frac{d(x_1 w_1 + x_2 w_2 + b1)}{dw_2} = x_2 = 6$$

$$\frac{dE}{dw_2} = -0.4750 \times 0.2494 \times 1 \times 0.04948 \times 6$$

$$= -0.0352$$

$$w_2^{new} = w_2^{old} - \alpha \frac{dE}{dw_2}$$

$$= -0.5 - (0.1)(-0.0352)$$

$$w_2^{new} = -0.4965$$

\# $$\frac{dE}{db_1} = \frac{dE}{da_4} \cdot \frac{da_4}{dz_4} \cdot \frac{dz_4}{da_1} \cdot \frac{da_1}{dz_1} \cdot \frac{dz_1}{db_1}$$

$$\frac{dz_1}{db_1} = \frac{d(x_1 w_1 + x_2 w_2 + b_1)}{db_1} = 1$$

$$\frac{dE}{db_1} = -0.4750 \times 0.2494 \times 1 \times 0.04948 \times 1$$

$$= -5.8616 \times 10^{-3}$$

$$b_1^{new} = b_1^{old} - \alpha \frac{dE}{db_1^{old}} = 0.1 - (0.1)(-5.8616 \times 10^{-3})$$

$$b_1^{new} = 0.1006$$

\# $$\frac{dE}{dw_3} = \frac{dE}{da_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_3}$$

$$\frac{\partial z_4}{\partial a_2} = \frac{\partial(n1 \cdot w7 + n2 \cdot w8 + n3 \cdot w9)}{da_2} = w8 \qquad a_2 = n_2$$

$$= -1$$

$$\frac{\partial a_2}{\partial z_2} = \frac{\partial(\sigma_2(z_2))}{\partial z_2} = \sigma_2(1-\sigma_2) \qquad \sigma_2 = a_2 = n_2$$

$$= 0.0522(1-0.0522)$$

$$= \underline{0.04948}$$

$$\frac{\partial z_2}{\partial w_3} = \frac{\partial(x_1 w_3 + x_2 w_4 + b_2)}{\partial w_3} = x_1 = 0$$

$$\therefore \frac{\partial E}{\partial w_3} = 0$$

$$w_3^{new} = w_3^{old} - \alpha \frac{\partial E}{\partial w_3} = 0.5 - 0$$

$$w_3^{new} = 0.5 \qquad = \underline{\underline{0.5}}$$

$$\frac{dE}{dw_4} = \frac{dE}{da_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_4}$$

$$\frac{\partial z_2}{\partial w_4} = \frac{\partial(x_1 w_3 + x_2 w_4 + b_2)}{\partial w_4} = x_2 = 6$$

$$\frac{dE}{dw_4} = -0.4750 \times 0.2494 \times -1 \times 0.04948 \times 6 = \underline{0.03517}$$

$$w_4^{new} = w_4^{old} - \alpha \frac{\partial z_2}{\partial w_4} = -0.5 - (0.1)(0.03517)$$

$$= \underline{-0.5035}$$

$$w_4^{new} = \underline{-0.5035}$$

$$\frac{dE}{db_2} = \frac{dE}{da_4} \cdot \frac{da_4}{\partial z_4} \cdot \frac{dz_4}{\partial a_2} \cdot \frac{da_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2}$$

$$\frac{\partial z_2}{\partial b_2} = \frac{\partial(x_1 w_3 + x_2 w_4 + b_2)}{\partial b_2} = 1$$

$$\therefore \frac{dE}{db_2} = -0.4750 \times 0.2494 \times -1 \times 0.04948 \times 1$$

$$= 5.8616 \times 10^{-3}$$

$$b^{new} = b^{old} - (0.1)(5.8616 \times 10^{-3}) = 0.099413$$

$$(0.1) - (0.1)(5.8616 \times 10^{-3}) = 0.099413$$

$b_2^{new} = 0.099414$

$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_5}$

$\frac{\partial z_4}{\partial a_3} = \frac{\partial(x1 \cdot w7 + x2 \cdot w8 + x3 \cdot w9 + b4)}{\partial a_3} = w9$

$= 0$ (If any of the term is 0, product = 0) $\because a_3 = x_3$

$\frac{\partial E}{\partial w_5} = 0$

$w_5^{new} = w_5^{old} - \alpha \frac{\partial E}{\partial w_5}$

$= 0.5 - 0.1 \times 0 = 0.5$

$w_5^{new} = 0.5$

$\frac{\partial E}{\partial w_6} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_6}$

$\frac{\partial E}{\partial w_6} = 0 \quad \because \frac{\partial z_4}{\partial a_3} = 0$

$w_6^{new} = w_6^{old} - \alpha\left(\frac{\partial E}{\partial w_6}\right) = -0.5$

$w_6^{new} = -0.5$

$\frac{\partial E}{\partial b_3} = \frac{\partial E}{\partial a_4} \cdot \frac{\partial a_4}{\partial z_4} \cdot \frac{\partial z_4}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial b_3}$

$\frac{\partial E}{\partial b_3} = 0 \quad \because \frac{\partial z_4}{\partial a_3} = 0 \quad \therefore b_3^{new} = b_3^{old} - \alpha \frac{\partial E}{\partial b_3^{old}} = 0.1$

Q1. B

Old parameters values	new parameter values
i/p $x_1 = 0, x = 6$	i/p $x_1 = 0, x = 6$
Parameters ⇒ OLD	Parameters ⇒ NEW (UPDATED)
$w_1 = 0.5$	$w_1' = 0.5$
$w_2 = -0.5$	$w_2' = -0.4965$
$w_3 = 0.5$	$w_3' = 0.5$
$w_4 = -0.5$	$w_4' = -0.5035$
$w_5 = 0.5$	$w_5' = 0.5$
$w_6 = -0.5$	$w_6' = -0.5$
$w_7 = 1$	$w_7' = 1.00062$
$w_8 = -1$	$w_8' = -0.9994$
$w_9 = 0$	$w_9' = 6.184 \times 10^{-4}$
$b_1 = 0.1$	$b_1' = 0.1006$
$b_2 = 0.1$	$b_2' = 0.099414$
$b_3 = 0.1$	$b_3' = 0.1$
$b_4 = 0.1$	$b_4' = 0.1118$

After the entire backpropagation is done, only then the weights are updated. Here, for ease of calculation, they have been done just after calculating the partial derivatives. As x1=0 there is no learning for w1, w3, and w5.

As w9 = 0, there is no contribution from w5 and w6 hence, no change in their weights.

same for b3 as w9=0, there is no contribution from b3 so it is not changed.

Homework 1 – [Bhati – 015309736]

Screenshots of Coding Output

My ID = 015309736. So the last two digits are 3 and 6.

```
# filter out the training and test sets with these two digits - 3 and 6
x_train = x_train[(y_train == 3) | (y_train == 6)]
y_train = y_train[(y_train == 3) | (y_train == 6)]
x_test = x_test[(y_test == 3) | (y_test == 6)]
y_test = y_test[(y_test == 3) | (y_test == 6)]
```

```
# For binary classification modify labels: Label encoding [3 => 0 and 6 => 1]
y_train = np.where(y_train == 3, 0, 1)
y_test = np.where(y_test == 3, 0, 1)
```

```
In [13]: # Pixel values range from 0 to 255. Normalize pixel values of train and test to the range [0, 1]
         x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
In [14]: x_train[0] # Normalized x_train
```

```
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.17647059, 0.87058824,
         0.98823529, 0.98823529, 0.98823529, 0.98823529, 0.99215686,
         0.98823529, 0.98823529, 0.98823529, 0.69411765, 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.17647059, 0.8745098 ,
         0.99215686, 0.99215686, 0.99215686, 0.99215686, 1.        ,
         0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.29019608,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.12156863,
         0.48235294, 0.20392157, 0.17254902, 0.17254902, 0.17254902,
         0.17254902, 0.56078431, 0.98823529, 0.98823529, 0.29019608,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
```

Neural Network for Binary Classification

[15]:
```
# Batch size and Learning rate are hyperparameters
batch_size = 32
learning_rate = 0.001
```

[16]:
```
model = Sequential()

model.add(Flatten(input_shape=(28,28))) ## 28x28 pixels are converted to 1D 784 input units.
model.add(Dense(128,activation='relu')) # 128 nodes with ReLU activation to add non-linearity.
model.add(Dense(1,activation='sigmoid')) ## 1 output node for binary classication using sigmoid.
```

[17]:
```
model.summary()

```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 100,609
Trainable params: 100,609
Non-trainable params: 0
_________________________________________________________________
```

Params in the second layer are 100480 because 784 inputs * 128 nodes(weights = 100352) + 128(baises) = 100480

Params in the last layer = 129 because 128(wts) + 1(bias) = 129

Binary_Crossentropy is used for binary classification and optimization is chosen as Adam

[18]:
```
# Compiling the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), loss='binary_crossentropy',metrics=['accuracy'
```

Early stopping criteria based on validation loss

[19]:
```
# Early stopping criteria based on validation loss
# patience = Number of epochs with no improvement after which training will be stopped.
# restore_best_weights = Whether to restore model weights from the epoch with the best value of the monitored quantity.
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

[20]:
```
# Training the model
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=20, validation_split=0.2, callbacks=[early_stopping])
```

```
Epoch 1/20
302/302 [==============================] - 1s 2ms/step - loss: 0.0344 - accuracy: 0.9899 - val_loss: 0.0086 - val_accuracy: 0.9
975
Epoch 2/20
302/302 [==============================] - 0s 1ms/step - loss: 0.0098 - accuracy: 0.9969 - val_loss: 0.0055 - val_accuracy: 0.9
979
Epoch 3/20
302/302 [==============================] - 0s 2ms/step - loss: 0.0054 - accuracy: 0.9985 - val_loss: 0.0045 - val_accuracy: 0.9
979
Epoch 4/20
302/302 [==============================] - 0s 2ms/step - loss: 0.0026 - accuracy: 0.9997 - val_loss: 0.0021 - val_accuracy: 0.9
996
Epoch 5/20
302/302 [==============================] - 0s 1ms/step - loss: 0.0011 - accuracy: 0.9999 - val_loss: 0.0082 - val_accuracy: 0.9
967
Epoch 6/20
```

30]:

```
# Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_accuracy)
```

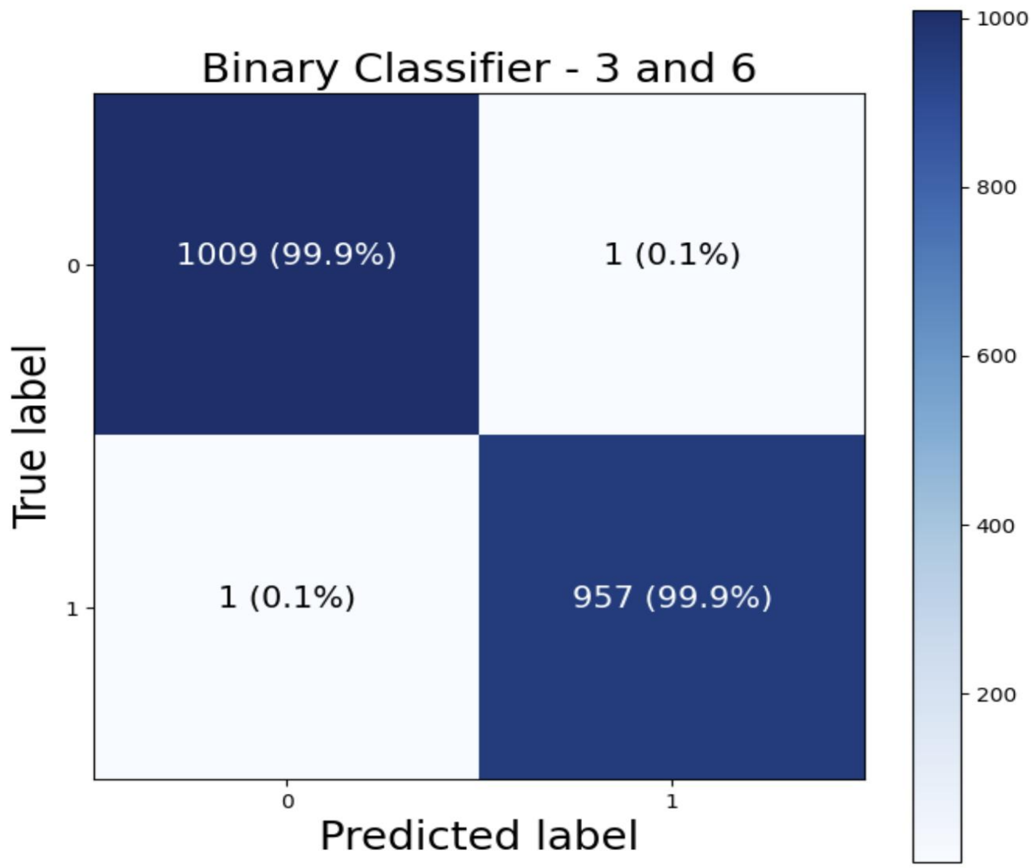
```
62/62 [==============================] - 0s 869us/step - loss: 0.0024 - accuracy: 0.9990
Test Accuracy: 0.9989837408065796
```

Construct a confusion matrix

28]:

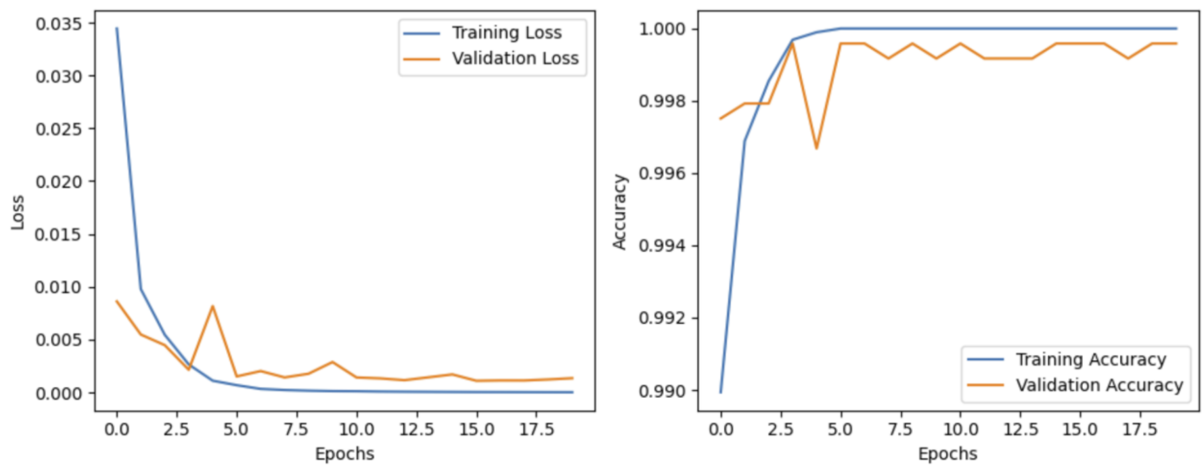
```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
draw_confusion_matrix(y_test, y_pred,"Binary Classifier - 3 and 6")
```

```
Confusion Matrix:
[[1009    1]
 [   1  957]]
```



Learning Curve - Loss and Accuracy

.]:
```python
# Present the learning curve
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```



Show some examples of predictions

:
```python

eg_indices = np.random.choice(len(x_test), 10) # 10 randomly selected samples
eg_images = x_test[eg_indices]
eg_labels = y_test[eg_indices]
eg_predictions = model.predict(eg_images)
for i in range(len(eg_images)):
    plt.figure()
    plt.imshow(eg_images[i], cmap='Greys')
    plt.title(f"True Label: {eg_labels[i]}, Predicted: {np.round(eg_predictions[i][0])}")
    plt.axis('off')
```

True Label: 1, Predicted: 1.0

True Label: 0, Predicted: 0.0

TABLE for 2-Class(Binary Class) Hyperparameters

Activation function – hidden layer	ReLU
Activation function - output layer	Sigmoid
Weight Initializer	Default -GlorotUniform
Number of hidden layers	1 (2 including output layer)
Neurons in hidden layers	128
Neurons in the output layer	1
Loss function	Binary_crossentropy
Optimizer	Adam
Number of epochs	20
Batch size	32
Learning rate	0.001
Evaluation Metric	Accuracy

B. Three different Weight Initializers. As per keras documentation there are multiple initializers such as He normal, GlorotNormal, RandomNormal, and GlorotUniform.

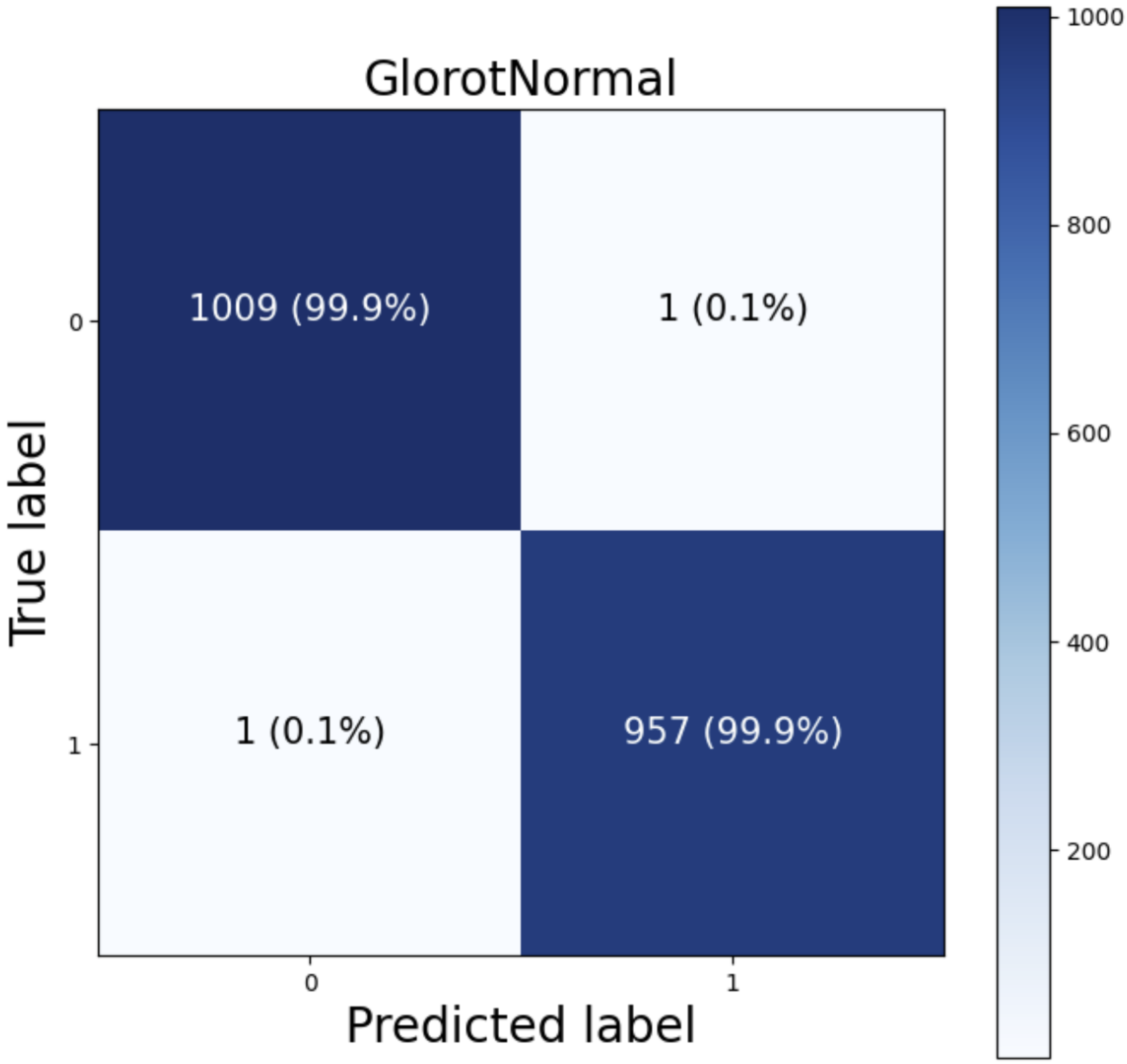
3]:
```
from tensorflow.keras.initializers import GlorotNormal, HeNormal, RandomNormal,GlorotUniform

```

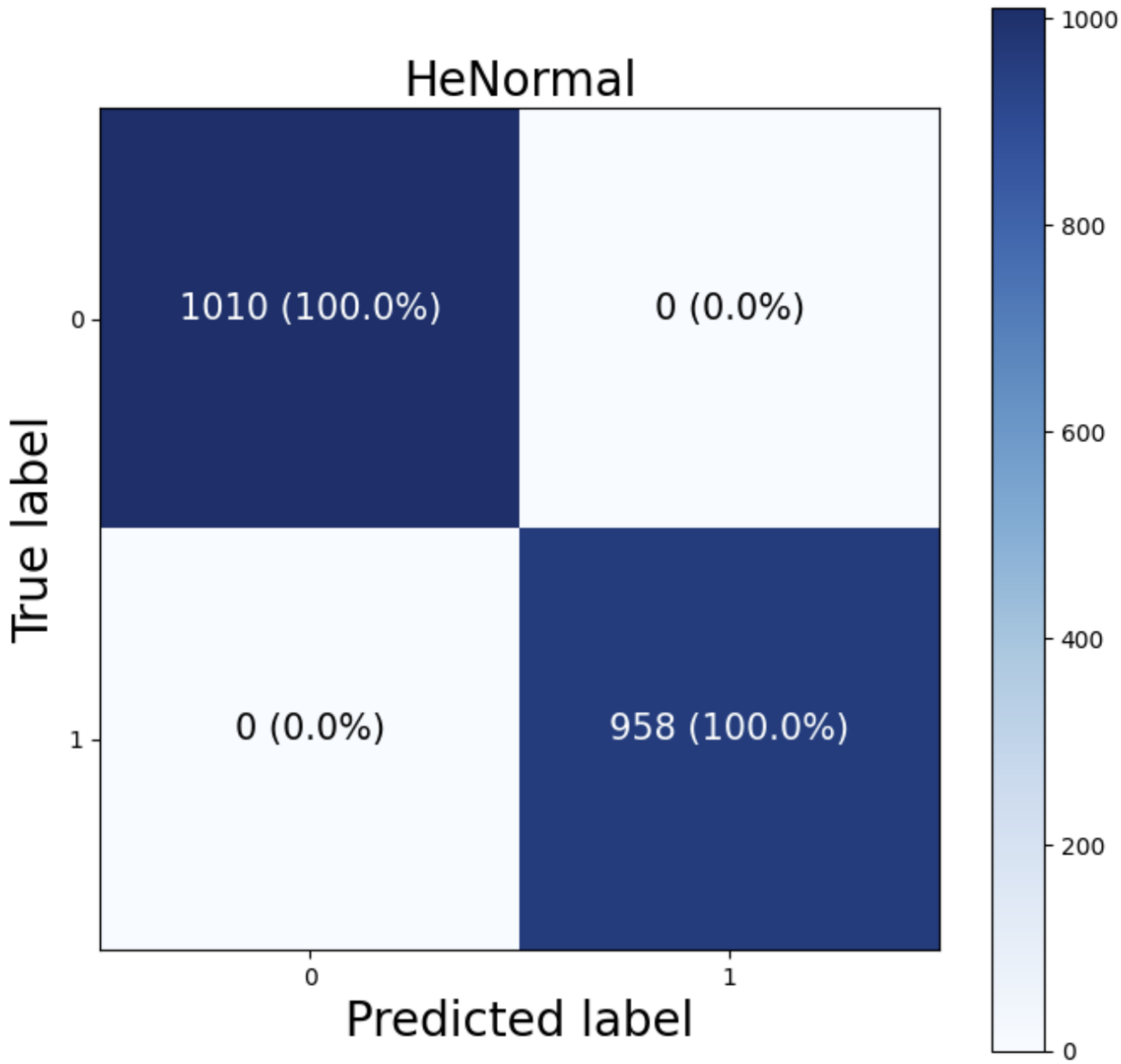
```
# Lists to store results
learning_curves = []
accuracies = []
```

```
labels = ["GlorotNormal", "HeNormal", "RandomNormal","GlorotUniform"]
i=0
# Build and train models for each initializer
for initializer in initializers:

    model = Sequential([
        Flatten(input_shape=(28, 28)),
        Dense(128, activation='relu', kernel_initializer=initializer),
        Dense(1, activation='sigmoid')
    ])
    model.summary()
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), loss='binary_crossentropy', metrics=['ac
     # Train the model
    history = model.fit(x_train, y_train, batch_size=batch_size, epochs=30, validation_split=0.2, verbose=0)

    # Evaluate on test data
    y_pred = np.round(model.predict(x_test))
    accuracy = accuracy_score(y_test, y_pred)
    confusion = confusion_matrix(y_test, y_pred)
    draw_confusion_matrix(y_test, y_pred,labels[i])

    #confusion_matrices.append(confusion)
    learning_curves.append(history.history)
    accuracies.append(accuracy)
    i=i+1
```

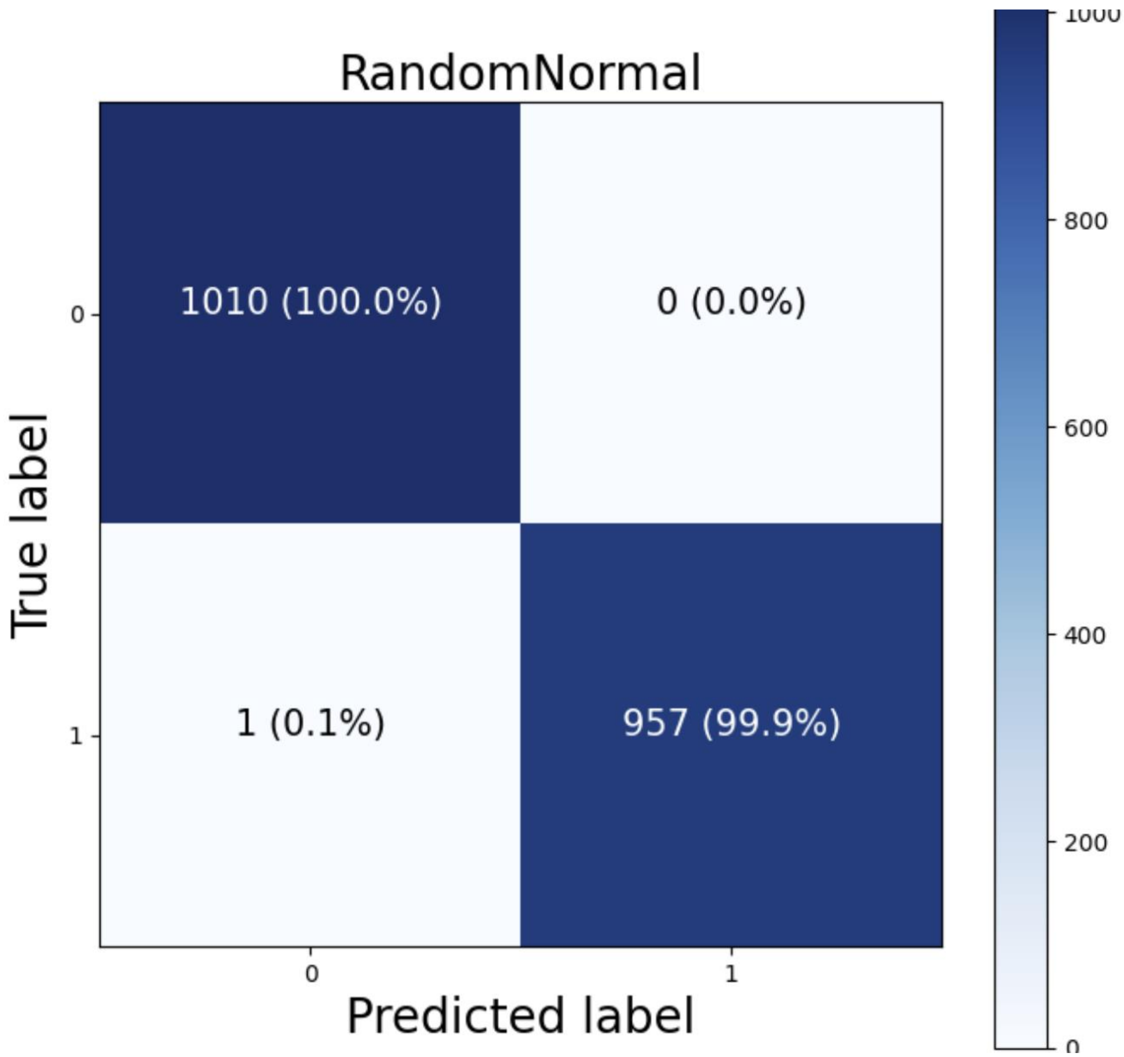
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 784)               0

 dense_2 (Dense)             (None, 128)               100480

 dense_3 (Dense)             (None, 1)                 129

=================================================================
Total params: 100,609
Trainable params: 100,609
Non-trainable params: 0
_________________________________________________________________
62/62 [==============================] - 0s 738us/step
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_2 (Flatten)         (None, 784)               0

 dense_4 (Dense)             (None, 128)               100480

 dense_5 (Dense)             (None, 1)                 129

=================================================================
Total params: 100,609
Trainable params: 100,609
Non-trainable params: 0
_________________________________________________________________
62/62 [==============================] - 0s 721us/step
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_3 (Flatten)         (None, 784)               0

 dense_6 (Dense)             (None, 128)               100480

 dense_7 (Dense)             (None, 1)                 129

=================================================================
Total params: 100,609
Trainable params: 100,609
Non-trainable params: 0
```

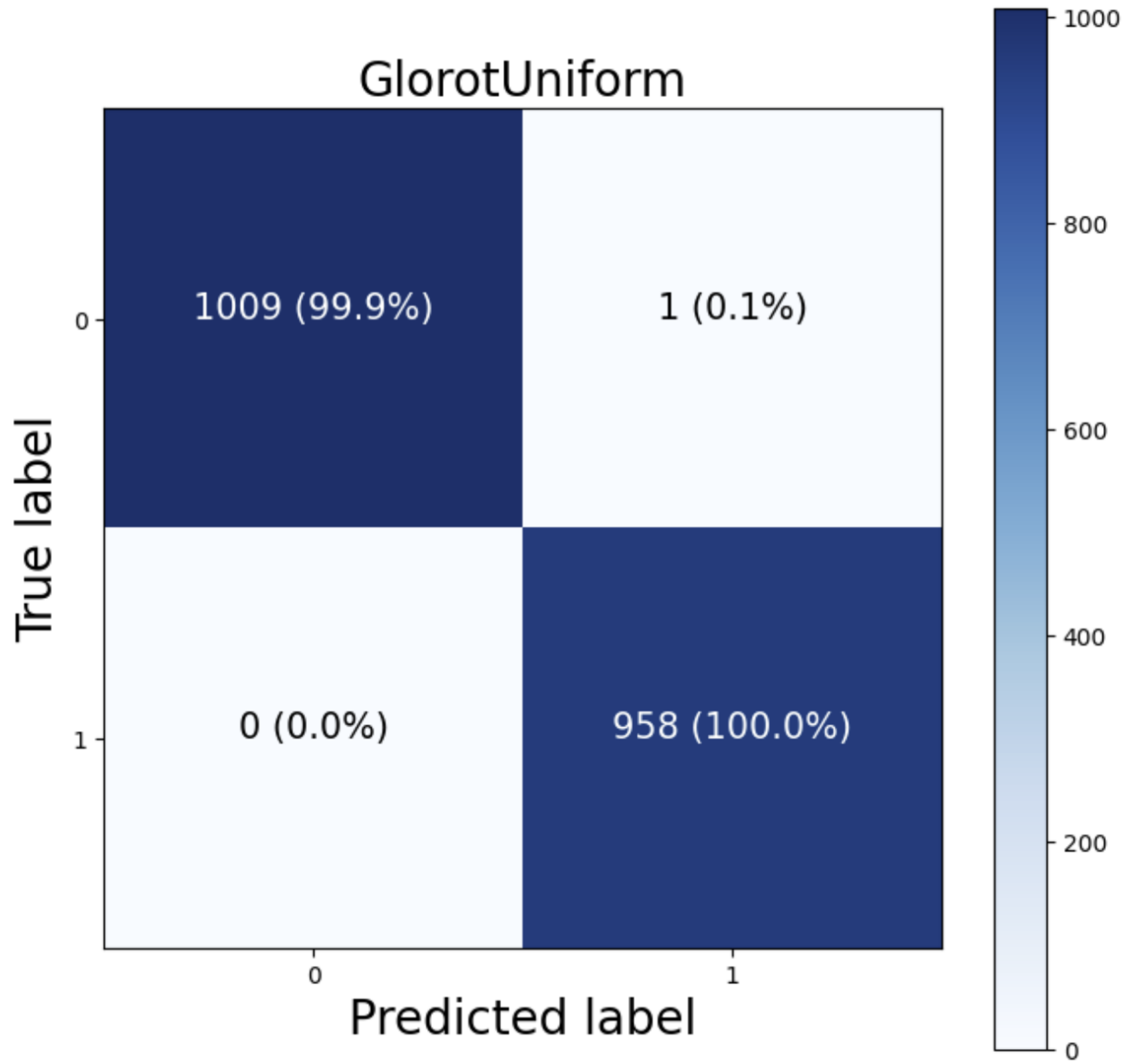
```
62/62 [==============================] - 0s 754us/step
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_4 (Flatten)         (None, 784)               0

 dense_8 (Dense)             (None, 128)               100480

 dense_9 (Dense)             (None, 1)                 129

=================================================================
Total params: 100,609
Trainable params: 100,609
Non-trainable params: 0
_________________________________________________________________
62/62 [==============================] - 0s 869us/step
```


GlorotNormal
1009 (99.9%)
1 (0.1%)
1 (0.1%)
957 (99.9%)
True label
Predicted label
0
1
0
1
1000
800
600
400
200



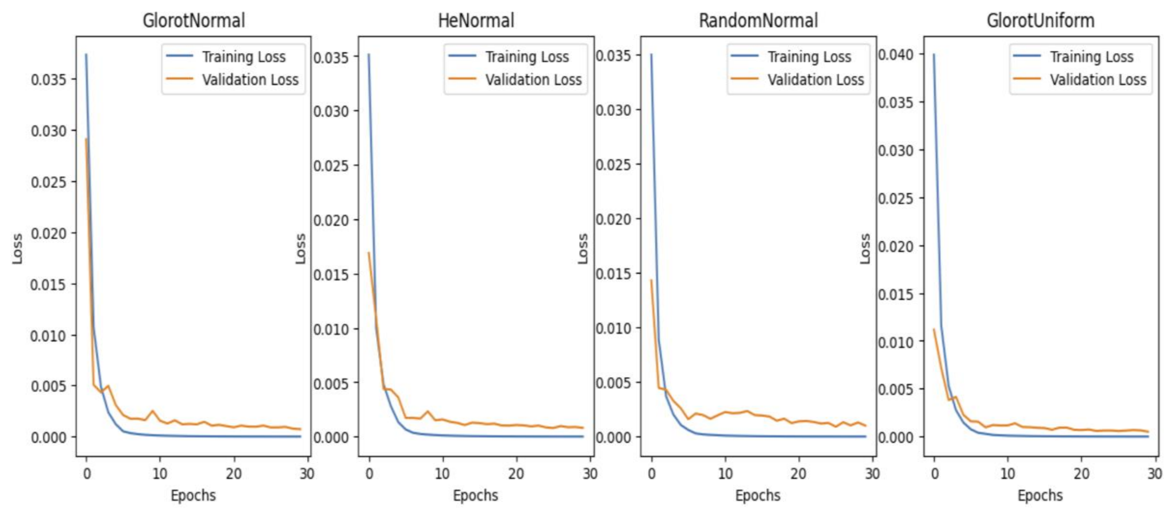
HeNormal
1010 (100.0%)
0 (0.0%)
0 (0.0%)
958 (100.0%)
True label
Predicted label
0
1
0
1
1000
800
600
400
200
0


RandomNormal
1010 (100.0%)
0 (0.0%)
1 (0.1%)
957 (99.9%)
True label
Predicted label
0
1
0
1
1000
800
600
400
200
0



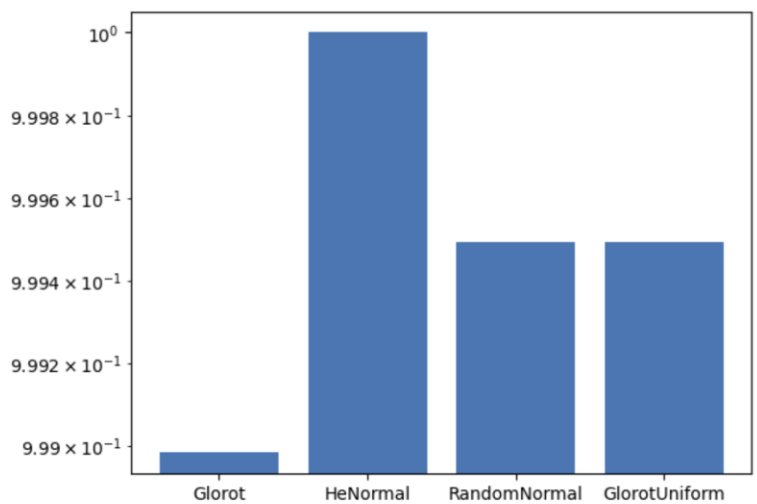
All of these initializers initializes the weights with a normal distribution centered around 0 but with difference in standard deviation. Glorot calculates standard deviation based on number of input units and output units, HE Normal calculates standard deviation with respect to number of input units and RandomNormal calculates std. as specified by the user. Here, He Normal performs the best with 100% accuracy. It performs better with ReLU activation function that has been used in the hidden layers.

3



HeNormal shows a less erractic behavior as compared to Glorot and RandomNormal.

HeNormal is also smoother after 10 epochs as compared to other initializers. RandomNormal is the most erratic throughout the epoch cycle.



In the above plot, HE Normal performs the best with the highest accuracy, followed by RandomNormal and GlorotUniform.

He Normal performs best with ReLU and Glorot performs best with sigmoid. Here, in the hidden layer ReLU is used, therefore, HeNormal performs better than the other.

TABLE for 2-Class Hyperparameters – With Four Different Weight Initializers

Activation function – hidden layer	ReLU
Activation function - output layer	Sigmoid
Weight Initializer	GlorotNormal(seed=1), HeNormal(seed=2), RandomNormal(mean=0.0, stddev=0.1, seed=3), GlorotUniform(seed=4
Number of hidden layers	1 (2 including the output layer)
Neurons in hidden layers	128
Neurons in the output layer	1
Loss function	Binary_crossentropy
Optimizer	Adam
Number of epochs	20
Batch size	32
Learning rate	0.001
Evaluation Metric	Accuracy

Q 3. 10 - class classification of MNIST

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_6 (Flatten)         (None, 784)               0

 dense_12 (Dense)            (None, 128)               100480

 dense_13 (Dense)            (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
```

]:

```
iling the model .We use sparse categorical that will hot encode the classes internally.
compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), loss='sparse_categorical_crossentropy',metrics=['acc
```

]:

```
# Early stopping criteria based on validation loss
# patience = Number of epochs with no improvement after which training will be stopped.
# restore_best_weights = Whether to restore model weights from the epoch with the best value of the monitored quantity.
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

]:

```
# Training the model
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=20, validation_split=0.2, callbacks=[early_stopping])
```

```
Epoch 1/20
1500/1500 [==============================] - 2s 1ms/step - loss: 0.2875 - accuracy: 0.9175 - val_loss: 0.1539 - val_accuracy:
0.9578
Epoch 2/20
1500/1500 [==============================] - 2s 1ms/step - loss: 0.1271 - accuracy: 0.9634 - val_loss: 0.1077 - val_accuracy:
0.9690
Epoch 3/20
1500/1500 [==============================] - 2s 1ms/step - loss: 0.0879 - accuracy: 0.9744 - val_loss: 0.0967 - val_accuracy:
```

5]:

```
# Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_accuracy)
```

```
313/313 [==============================] - 0s 772us/step - loss: 0.0789 - accuracy: 0.9781
Test Accuracy: 0.9781000018119812
```

CONFUSION MATRIX

[67]:

```
#draw_confusion_matrix(y_test, y_pred_classes,"classes")
# Construct a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred_classes)
print("Confusion Matrix:")
print(conf_matrix)
```

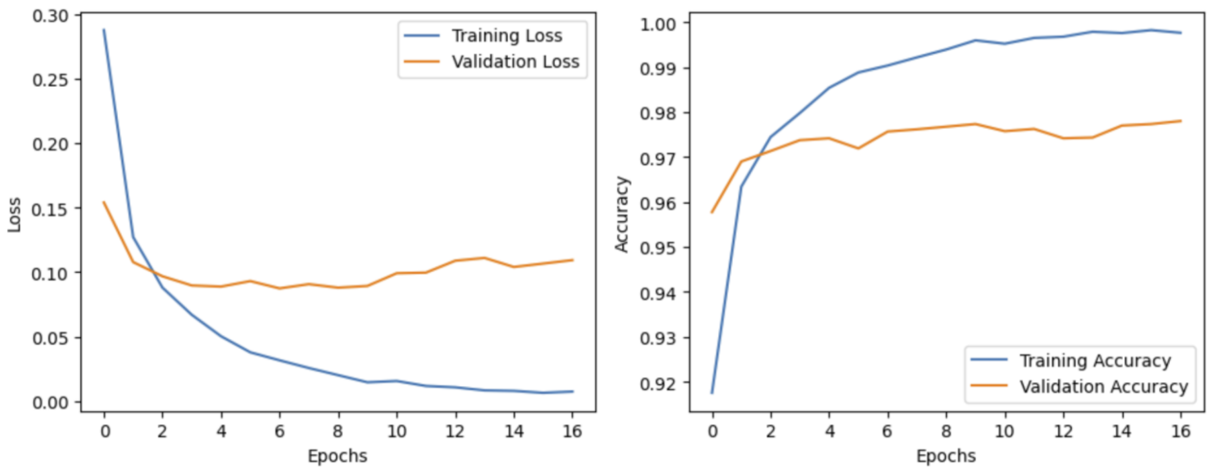
```
Confusion Matrix:
[[ 969    0    1    1    1    1    3    2    2    0]
 [    0 1120    4    2    0    1    3    1    4    0]
 [    2    0 1014    5    1    0    2    5    3    0]
 [    0    0    5  996    0    1    0    3    3    2]
 [    1    0    3    0  961    0    5    2    0   10]
 [    2    0    0   17    1  859    6    2    4    1]
 [    6    2    3    1    3    1  939    0    3    0]
 [    1    3   12    4    0    0    0 1002    3    3]
 [    3    0    5   11    4    1    1    4  942    3]
 [    1    3    0    6    7    3    0    9    1  979]]
```

As per the above confusion matrix, mostly all the classes from 0-9 are classified correctly. Class 0 and 1 are the least misclassified. Class 5 is misclassified as class 3 for 17 times; Class 7 has been misclaasified as 2 for 12 times; Class 8 is misclassified as 3 for 11 times.

```python
# Display a classification report
class_report = classification_report(y_test, y_pred_classes)
print("Classification Report:")
print(class_report)
```

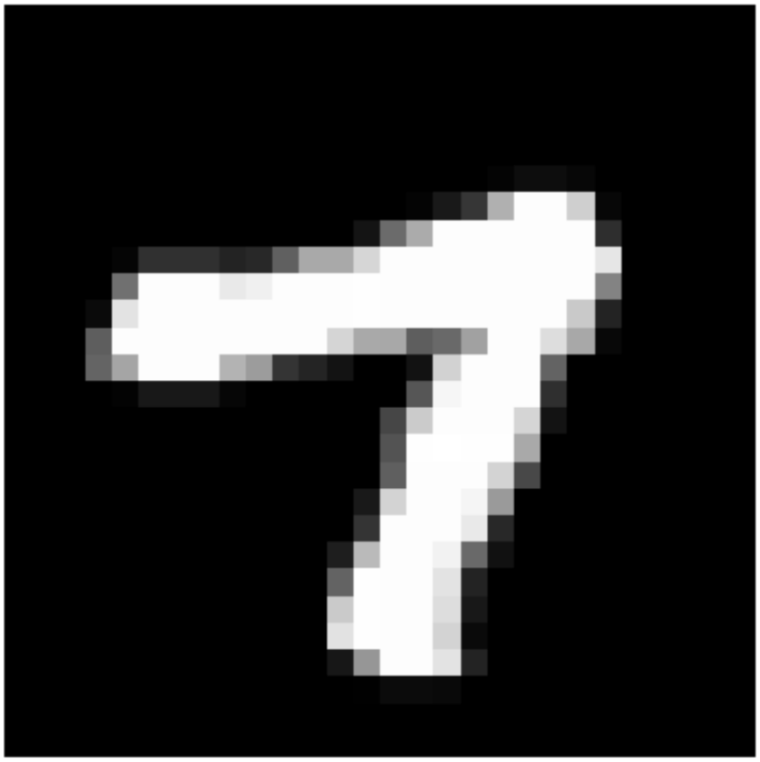
```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.98      0.98      1032
           3       0.95      0.99      0.97      1010
           4       0.98      0.98      0.98       982
           5       0.99      0.96      0.98       892
           6       0.98      0.98      0.98       958
           7       0.97      0.97      0.97      1028
           8       0.98      0.97      0.97       974
           9       0.98      0.97      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

```
# Plot the learning curve
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```




```
# Show some examples of predictions
eg_indices = np.random.choice(len(x_test), 10) # 10 randomly selected samples
eg_images = x_test[eg_indices]
eg_labels = y_test[eg_indices]
eg_predictions = model.predict(eg_images)
eg_predictions_classes = np.argmax(eg_predictions, axis=1)
for i in range(len(eg_images)):
    plt.figure()
    plt.imshow(eg_images[i], cmap='gray')
    plt.title(f"True Label: {eg_labels[i]}, Predicted: {eg_predictions_classes[i]}")
    plt.axis('off')

```

True Label: 7, Predicted: 7



True Label: 2, Predicted: 2

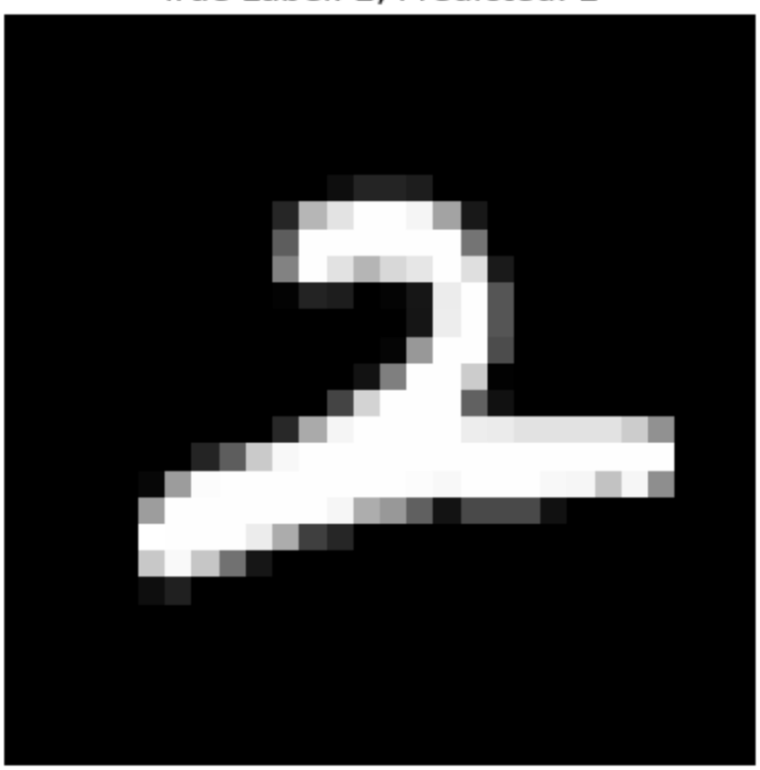


TABLE for 10 Class hyperparameters

Activation function – hidden layer	ReLU
Activation function - output layer	Softmax
Weight Initializer	Default - GlorotUniform
Number of hidden layers	1 (2 including the output layer)
Neurons in hidden layers	128
Neurons in the output layer	10
Loss function	sparse_categorical_crossentropy
Optimizer	Adam
Number of epochs	20
Batch size	32
Learning rate	0.001
Evaluation Metric	Accuracy