Which of the following techniques can help address the vanishing gradient problem when using sigmoid activation? [5 marks]

- a. Using more layers in the network
- b. Implementing batch normalization
- c. Increasing the batch size
- d. Using dropout regularization

normalize inputs to mean 0, variance 1
speed up training, helps gradients flow easily
solution for vanishing gradient problem

more layers - worsen vanishing gradient issue
when network gets deeper gradients deminish further

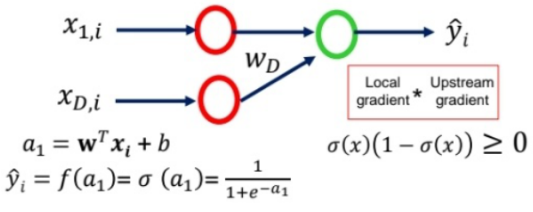
increase batch size - improve stability of training
improve convergence

dropout regularization - prevent overfitting

Vanishing gradients

Sigmoid $\sigma(x)$

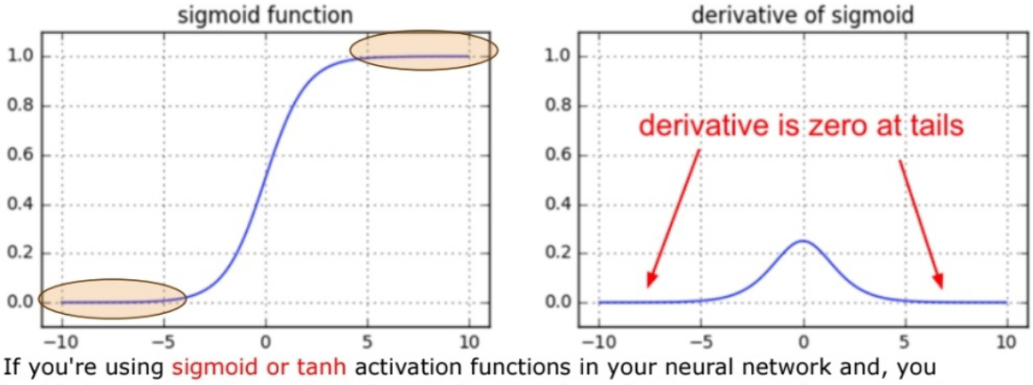
$\sigma'(x) = \sigma(x)(1-\sigma(x))$



- If we initialize **w** to be large
- $\sigma(x) \approx 1$ *or* $\sigma(x) \approx 0$
- $(1-\sigma(x)) \approx 0$ *or* $\sigma(x) \approx 0$
- $\sigma'(x) \approx 0$

The vanishing gradient problem occurs when gradients become extremely small as they flow backward through the layers of a neural network, causing early layers to learn very slowly or not at all because their weights are hardly updated.

Vanishing gradients



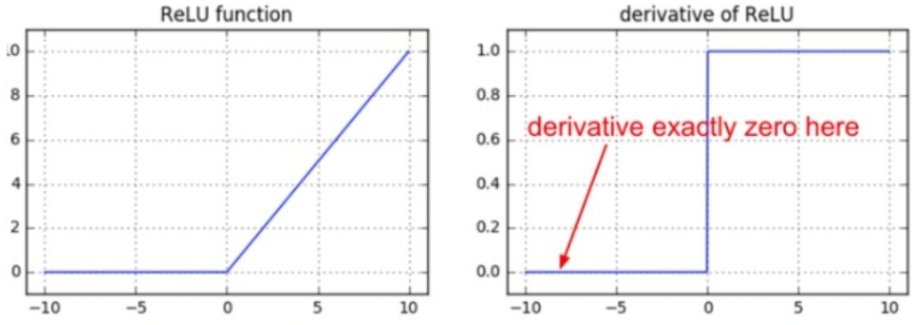
If you're using sigmoid or tanh activation functions in your neural network and, you should always be cautious about the initialization of weights to prevent these activations from becoming fully saturated.

Why does the Leaky ReLU function help address the vanishing gradient problem?

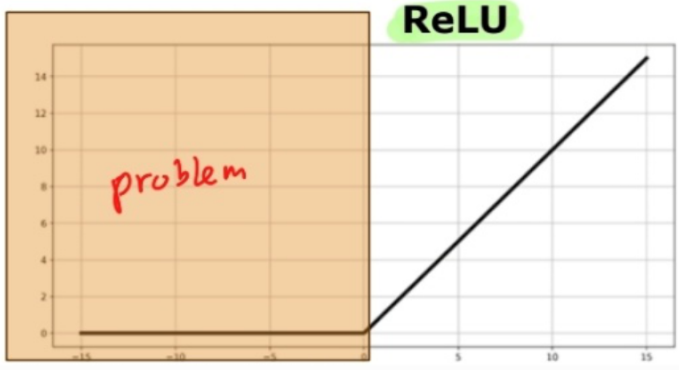
- a. It ensures all gradients are positive
- b. It simplifies the network architecture.
- c. It increases the number of parameters in the model.
- d. It prevents the activation from becoming zero for negative inputs, allowing gradients to flow during backpropagation.

Vanishing gradients

- ReLu solve this to some extend. It does not saturate for positive values.
- Computationally efficient (thresholding).

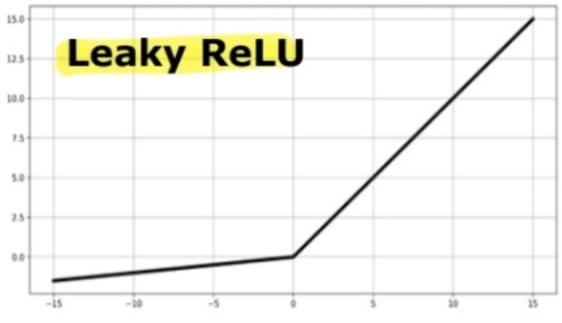


- Dying ReLUs: NN get stuck during training and always output zero, effectively becoming inactive neurons. No gradient update.
- Not zero centered



- Recommended for most cases

- Not saturate
- Neurons not die
- Computational efficient



$$\mathrm{LReLU}(x,\alpha) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{otherwise} \end{cases}$$

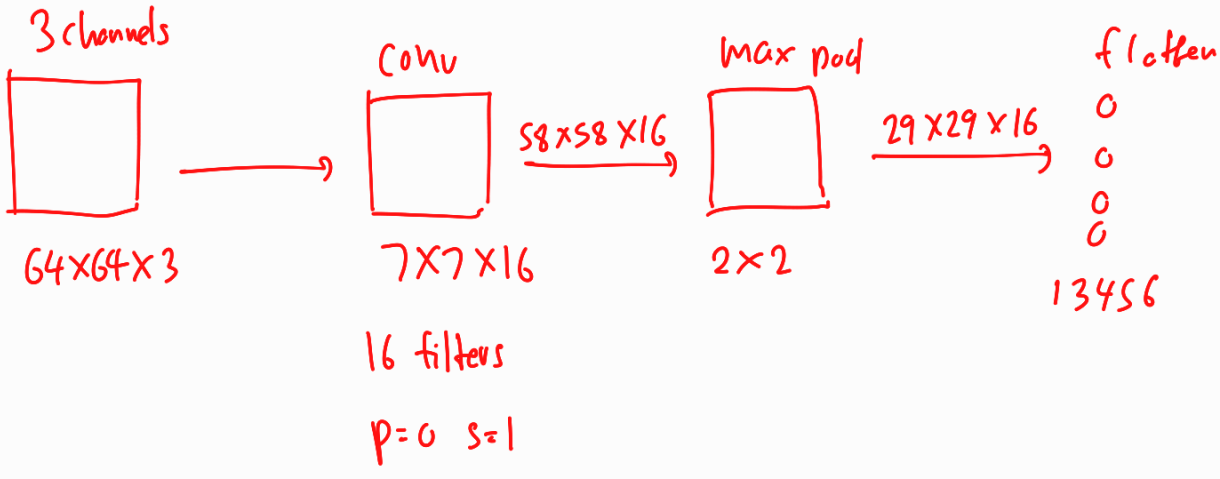
Parametric Rectified Linear Unit (PReLU)

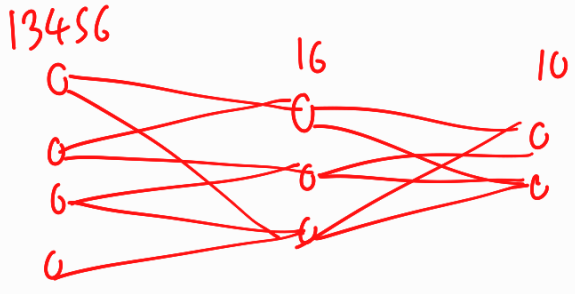
α is a learnable parameter that can be adjusted during training through backpropagation.

Consider a convolutional neural network (CNN) implementation for image classification. There are ten classes to recognize, and each input image is RGB image of size 64 × 64 pixels. The given CNN starts with a 7 × 7 convolutional layer containing 16 filters. This convolutional layer has no padding and stride is 1. Following this, there is a max-pooling layer with a 2 × 2 pooling window. After that feature maps are flattened, they are connected to a hidden layer with 16 neurons. These 16 nodes densely connect to the output nodes. How many learnable parameters (weights and biases)?

[10 marks]

- a. 100965
- b. 217882
- c. 216282
- d. 217850





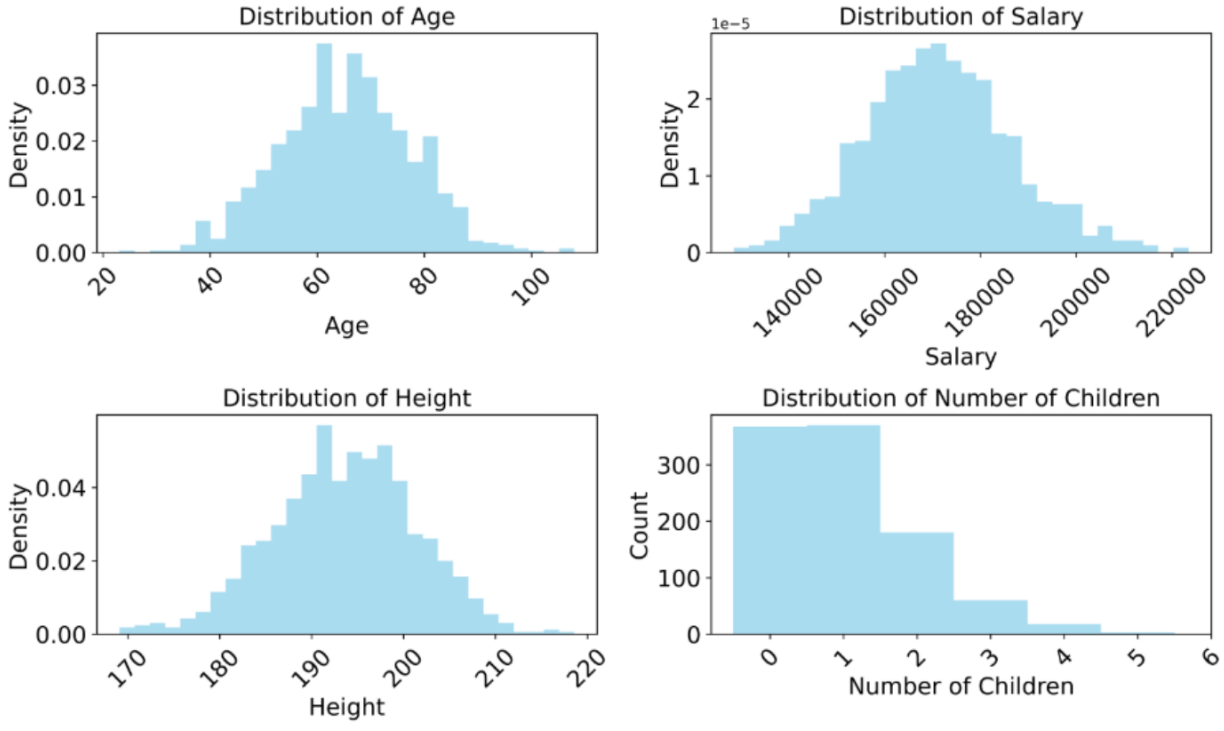
conv → $(7 \times 7 \times 3 + 1) \times 16 \rightarrow 2368$

flatten → $(13456 \times 16) + 16 \rightarrow 215312$

dense → $(16 \times 10) + 10 \rightarrow 170$

total → 217850

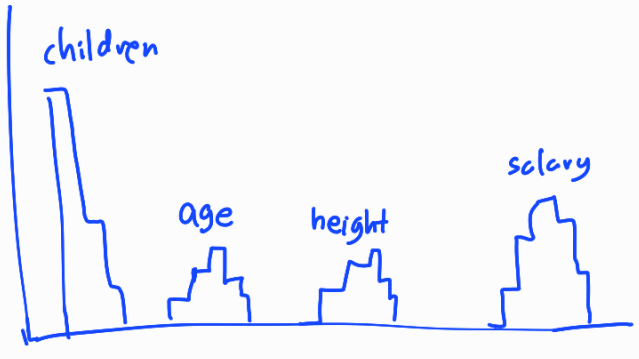
Suppose you are training a fully connected neural network for a regression task. Here, input features are 'Age', 'Salary', 'Height', and 'Number of Children'. This network has one hidden layer and ReLU function is used as the activation function. The distribution of different features are given in the below figure. Suppose that you do not use any normalization methods. What first order algorithm is most suitable to train this network ? **[5 marks]**



a. Gradient Descent

b. RMSprop

c. Gradient Descent with Momentum



gradient descent - slow convergence due to inefficient gradients

RMS prop - adapt learning rate for each parameter

scale down step size - large gradient parameters

scale up step size - small gradient parameters

gradient descent with momentum - accelerate convergence by speed up in constant gradient regions

What is one of the primary advantages of using depth-wise convolution in neural networks? [5 marks]

a. It increases the number of parameters significantly.

b. It always improves the model accuracy.

c. It reduces the computational cost and memory usage.

d. It requires larger amounts of labeled data for training.

depth wise convolution - reduce computational cost and memory usage

What happens when the stride is increased in a convolutional layer? **[5 marks]**

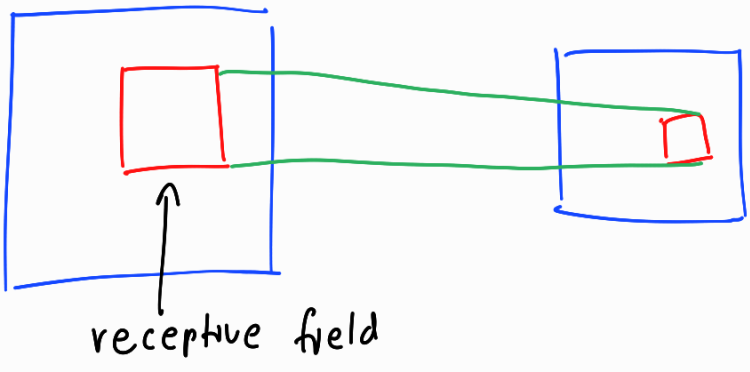
a. The output feature map size increases, while the receptive field size remains the same.

b. The receptive field size per output neuron increases, allowing each neuron to aggregate information from a larger portion of the input, but the output feature map size decreases.

c. There is no effect on the receptive field or the size of the output feature map.

d. The receptive field size per output neuron decreases, leading to finer details in the feature map.

Clear my choice



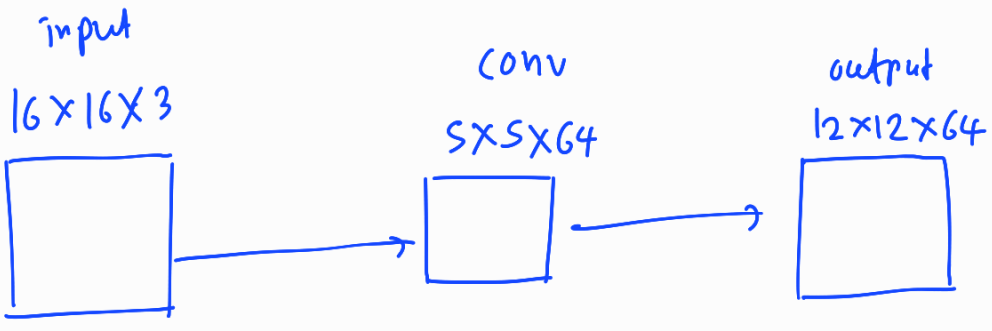
Suppose your input is 16 x 16 RGB image. Consider a kernel of 5 x 5 and the number of output channels as 64. Choose the correct answer. **[10 marks]**

a.
- Regular convolution number of multiplication operations for 100 images is 480000
- Depth-wise separable convolution number of multiplication operations for 100 images is 17100
- Improvement factor of Depth-wise separable convolution over Regular convolution is 28.07

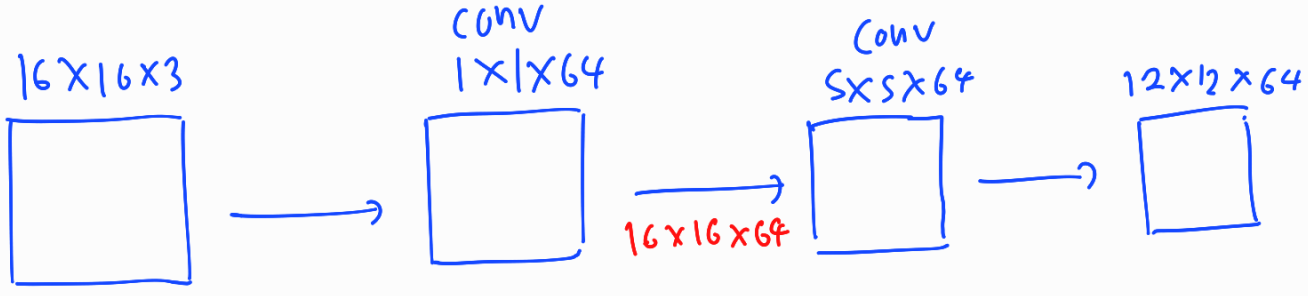
b.
- Regular convolution number of multiplication operations for 100 images is 48000000
- Depth-wise separable convolution number of multiplication operations for 100 images is 1710000
- Improvement factor of Depth-wise separable convolution over Regular convolution is 28.07

c.
- Regular convolution number of multiplication operations for 100 images is 69120000
- Depth-wise separable convolution number of multiplication operations for 100 images is 2462400
- Improvement factor of Depth-wise separable convolution over Regular convolution is 28.07

d.
- Regular convolution number of multiplication operations for 100 images is 691200
- Depth-wise separable convolution number of multiplication operations for 100 images is 24624
- Improvement factor of Depth-wise separable convolution over Regular convolution is 28.07



regular conv - $3 \times 64 \times 12 \times 12 \times 5 \times 5 \rightarrow 691200$

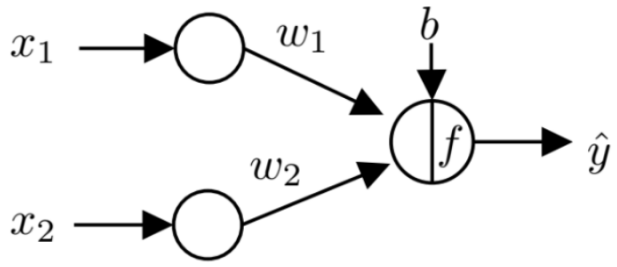


depth wise conv - $3 \times 1 \times 1 \times 16 \times 16 \times 64 \rightarrow 49152$

$- \quad 64 \times 1 \times 1 \times 5 \times 5 \times 64 \rightarrow 102400$

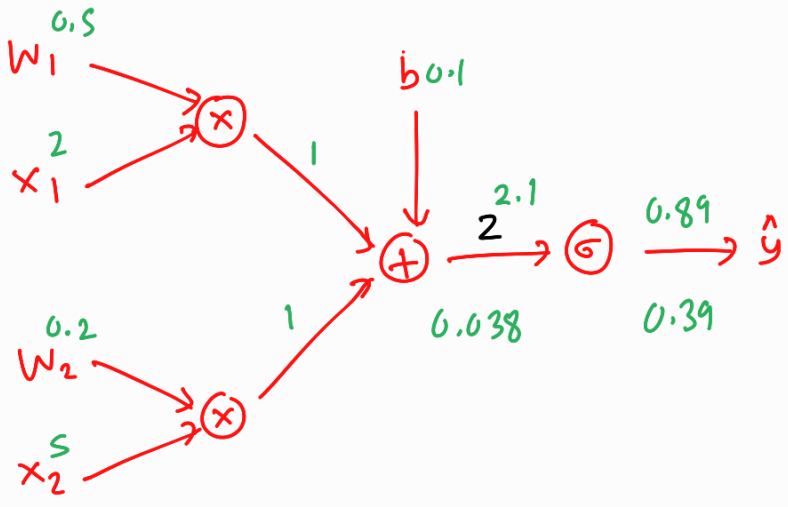
151552

?



Simple neural network.

Consider a simple neural network shown in Figure. Here, $f(\cdot)$ is the sigmoid activation function. For this neural network, the MSE loss function $L(\boldsymbol{w}, b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ is used. Here, y_i and $\hat{y}_i$ are the true value and output of the neural network, respectively. Suppose that we are considering only a one data sample ($y_i = y$ and $\hat{y}_i = \hat{y}$). Starting with initial values of $w_1 = 0.5$, $w_2 = 0.2$, $b = 0.1$, $y = 0.5$, $x_1 = 2$, and $x_2 = 5$, perform two iteration of gradient descent with a momentum of $m = 0.5$ and learning rate of 0.1. Suppose that initial velocities for w_1 and w_2 are set to zero ($v_1 = 0$ and $v_2 = 0$). What will be the updated value of w_2 after two iterations? [10 marks]



$$z = w_1 x_1 + w_2 x_2 + b = 2.1$$

$$\hat{y} = \sigma(2.1) = 0.89$$

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

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = 0.89 - 0.5 = 0.39$$

$$\frac{\partial \hat{y}}{\partial z} = 0.39 \times 0.89 \times 0.11 = 0.038$$

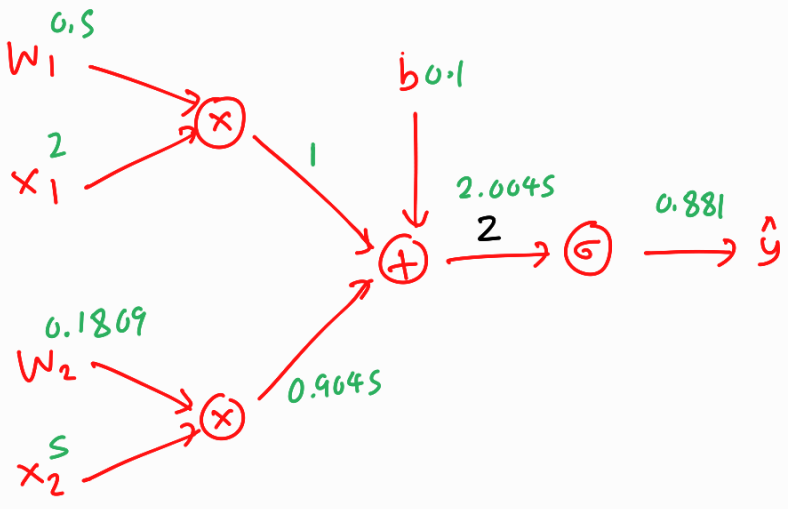
$$\frac{\partial z}{\partial w_2} = 5 \times 0.038 = 0.191$$

$$v_2 = m v_2 - \eta \frac{\partial z}{\partial w_2}$$

$$v_2 = (0.5)(0) - (0.1)(0.191) = -0.0191$$

$$w_2 = w_2 + v_2$$

$$w_2 = 0.2 - 0.0191 = 0.1809$$



$$z = w_1 x_1 + w_2 x_2 + b = 2.0045$$

$$\hat{y} = G(2.0045) = 0.881$$

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

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = 0.881 - 0.5 = 0.381$$

$$\frac{\partial \hat{y}}{\partial z} = 0.381 \times 0.881 \times 0.119 = 0.04$$

$$\frac{\partial z}{\partial w_2} = 5 \times 0.04 = 0.2$$

$$V_2 = mv_2 - \eta \frac{\partial 2}{\partial w_2}$$

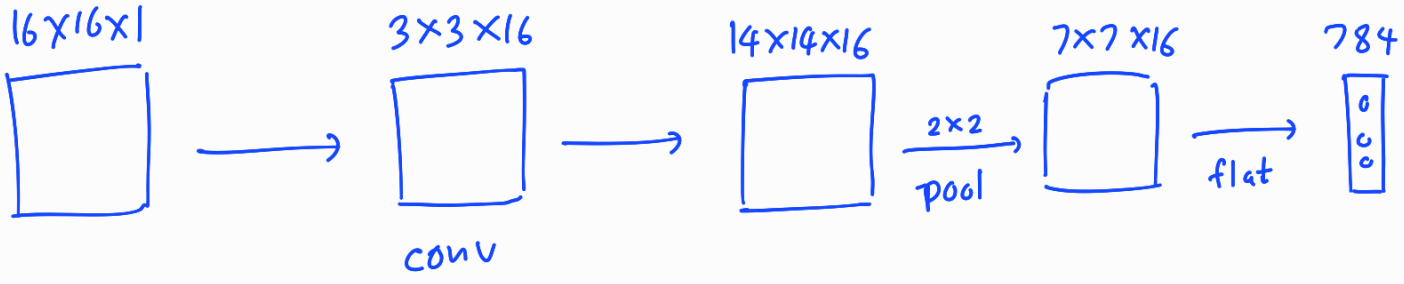
$$V_2 = (0.5)(-0.0191) - (0.1)(0.2) = -0.03$$

$$W_2 = W_2 + V_2$$

$$W_2 = 0.1809 - 0.03 = 0.1509$$

Consider a convolutional neural network (CNN) implementation for image classification. There are three classes to recognize, and each input image is grayscale and has a size of 16 × 16 pixels. The given CNN starts with a 3 × 3 convolutional layer containing 16 filters. This convolutional layer has no padding and stride is 1. Following this, there is a max-pooling layer with a 2 × 2 pooling window. After that feature maps are flattened, they are connected to a fully connected dense neural network (FCNN). This FCNN contains one hidden layer with 64 neurons with a Relu activation and then output layer with a Softmax activation. Compute input size (flattened feature maps size) to this fully connected dense neural network and output layer size. **[5 marks]**

- a. Input size is 1352 and output size is 5
- b. Input size is 784 and output size is 3
- c. Input size is 3136 and output size is 5
- d. Input size is 1152 and output size is 5



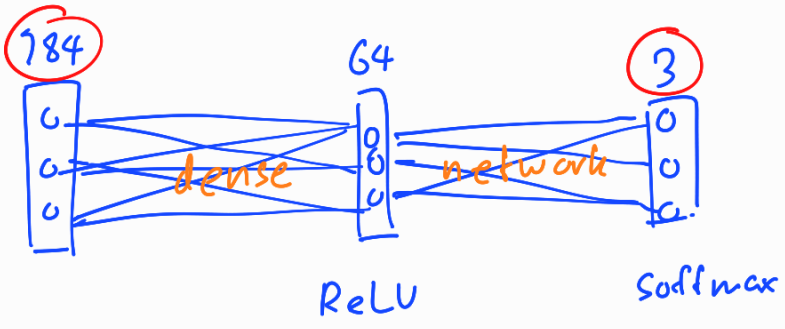


Figure shows an input data sample and a CNN weights. Apply the convolution operation on the input data sample using the given CNN weights. Note that the bias is set to 3. Here, both padding and stride are equal to 1. What is the output after the convolution operation ? Suppose that Leaky ReLU activation function with $\alpha = 0.001$ is applied on this output. What is the output after the Leaky ReLU activation? **[10 marks]**

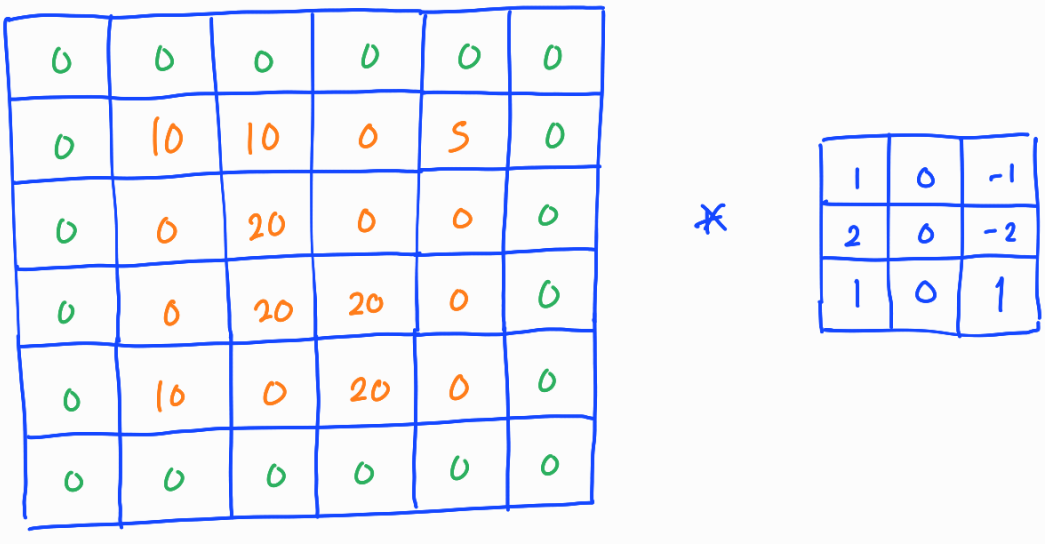
$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

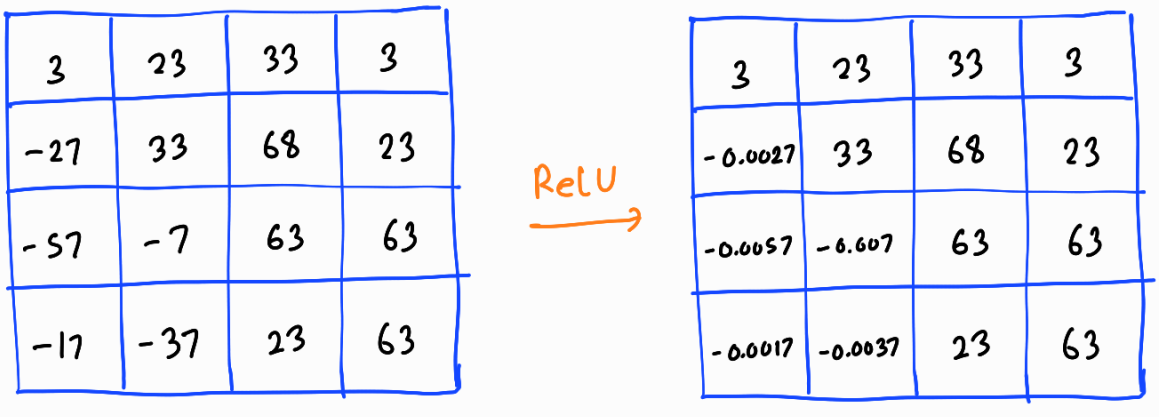
10	10	0	5
0	20	0	0
0	20	20	0
10	0	20	0

Input data sample.

1	0	-1
2	0	-2
1	0	1

CNN weights.





Following table shows training results of the SVM classifier for binary classification.

Sample	**Value** (x_1, x_2)	**Class Label**
Support Vector	(-2, 7)	-1
Support Vector	(4, 4)	1
Non-Support Vector	(-1 9)	-1
Non-Support Vector	(5, 2)	1
Non-Support Vector	(3, 2)	1
Test Sample	(-2 3)	y

Suppose that $w_0 = 0.47$, select the predicted class label y for the test sample.

$W_0 = 0.47$

$(-2,7) \rightarrow -1 \times \langle (-2,3), (-2,7) \rangle = -25$

$(4,4) \rightarrow 1 \times \langle (-2,3), (4,4) \rangle = 4$

decision value $= 0.47 - 25 + 4 = -20.53$

$-20.53 < 0 \rightarrow$ class -1

Time left 0:08:15

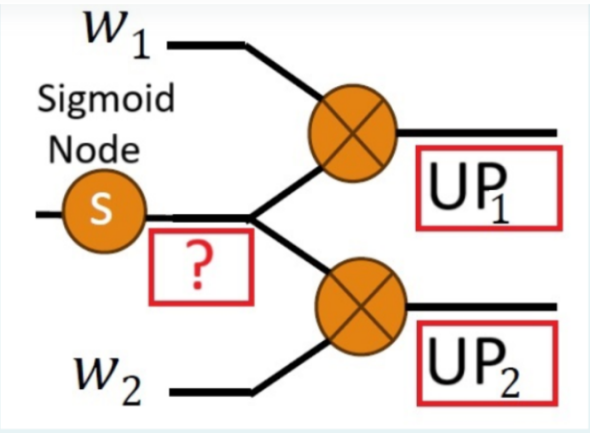
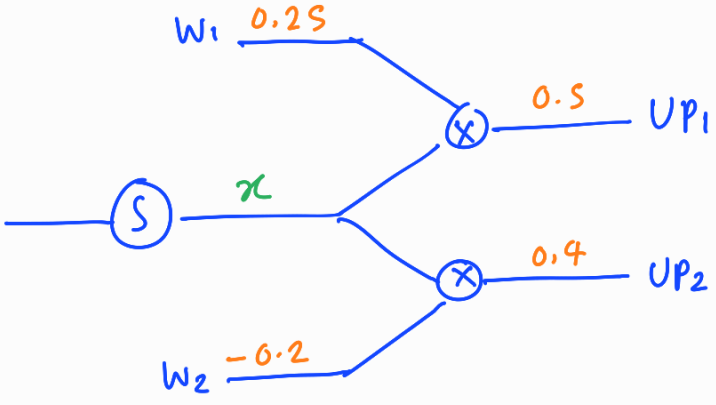


Figure shows portion of a neural network. Consider the sigmoid node and suppose that upstream gradients are given by UP_1 and UP_2. It is given that $w_1 = 0.25$ and $w_2 = -0.20$, $UP_1 = 0.5$ and $UP_2 = 0.4$. What is the gradient at the output of the Sigmoid node. [5 marks]

- a. -0.080
- b. 0.0224
- c. 0.125
- d. 0.0449



$x = -0.2 \times 0.4 + 0.25 \times 0.5 = 0.045$

The SVM loss function for a binary classification problem can be written as

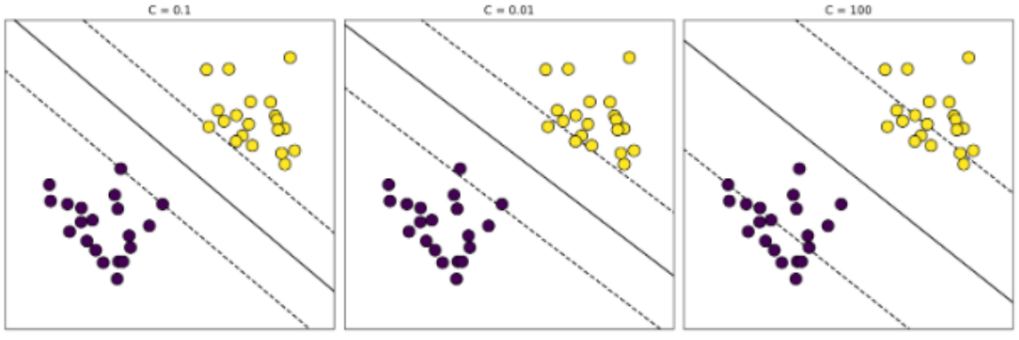
$$\min_{\tilde{\boldsymbol{w}},b} \frac{1}{2}\|\tilde{\boldsymbol{w}}\|_2^2 + \frac{1}{C}\sum_{i=1}^{N} \max(0, 1 - y_i(b + \tilde{\boldsymbol{w}}^T \boldsymbol{x}_i)).$$

Description of the notations are given as follows.

- $\tilde{\boldsymbol{w}}$: The weight vector and b: The bias term (intercept).
- C: A regularization parameter and N: The number of training samples.
- y_i: The true label for the i-th sample (+1 or −1).
- $\boldsymbol{x}_i$: The feature vector for the i-th sample and $\|\cdot\|_2$: L_2 norm.

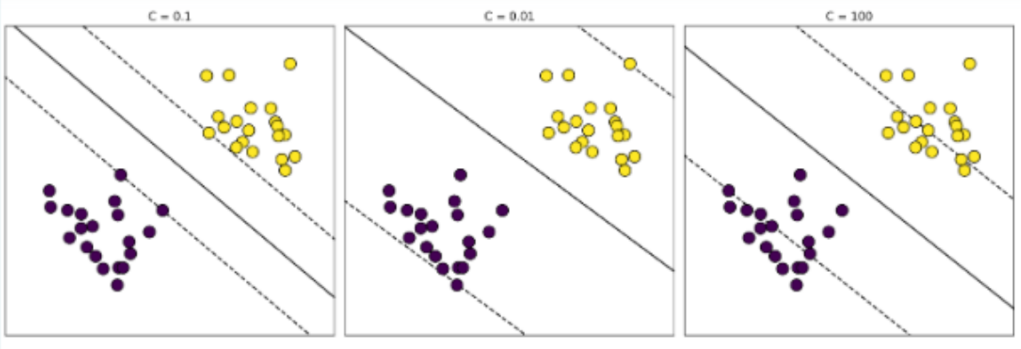
From the given figures, select the correct SVM decision boundaries corresponding to the C values of 0.1, 0.01 and 100. **[5 marks]**

a.



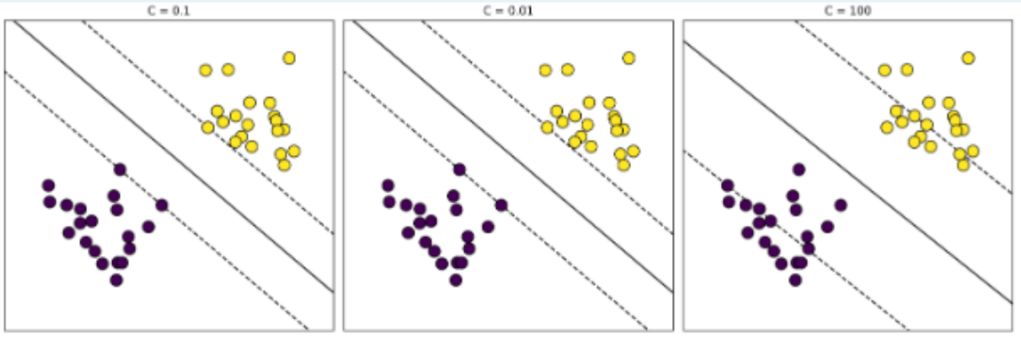
$C = 0.1$(left), $C = 0.01$(middle) and $C = 100$(right).

b.



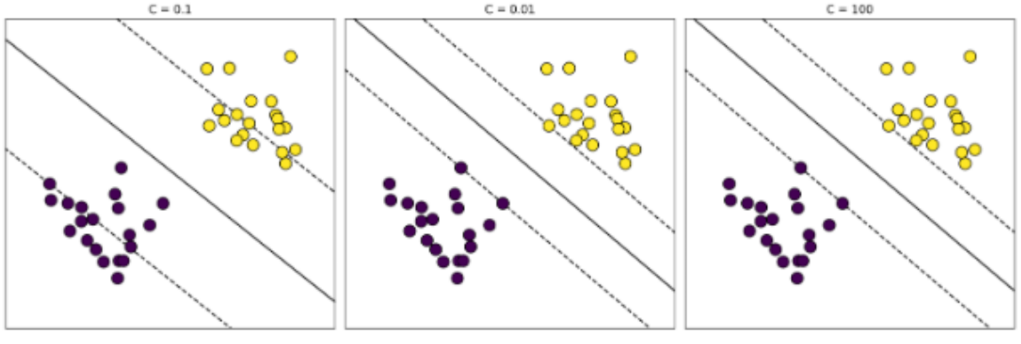
$C = 0.1$(left), $C = 0.01$(middle) and $C = 100$(right).

c.



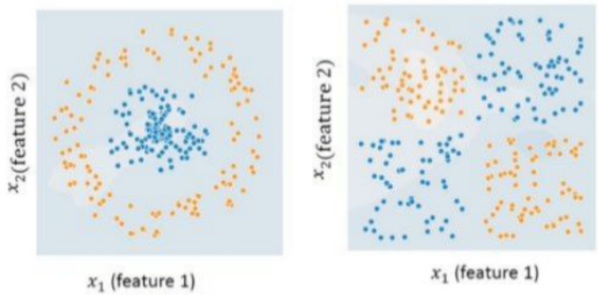
$C = 0.1$(left), $C = 0.01$(middle) and $C = 100$(right).

d.

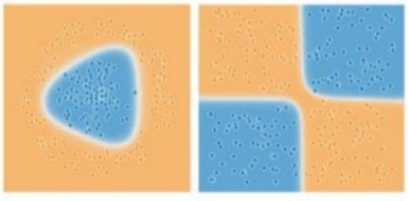


$C = 0.1$(left), $C = 0.01$(middle) and $C = 100$(right).

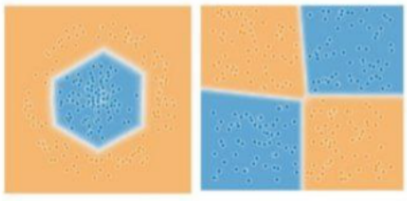
Clear my choice



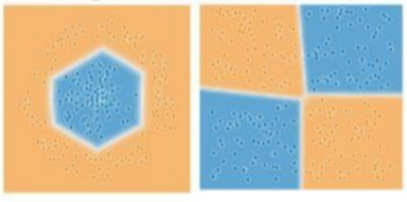
Q5. Figure shows two datasets which contain two features namely x_1 and x_2. These datasets belong to two distinct classes, represented by orange and blue colors. We will use a simple neural network with one hidden layer to classify these data points. What would be the decision boundaries for RelU and Sigmoid activation functions. **[5 marks]**



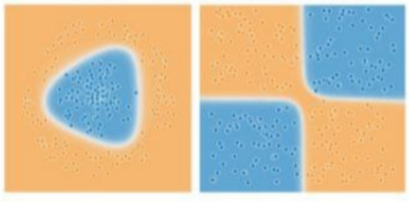
i. Sigmoid activation function



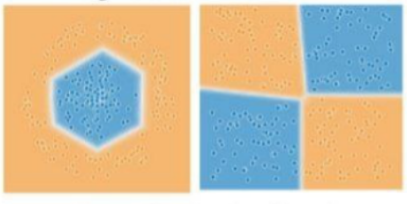
ii. ReLU activation function



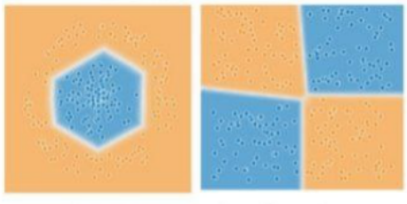
iii. Sigmoid activation function



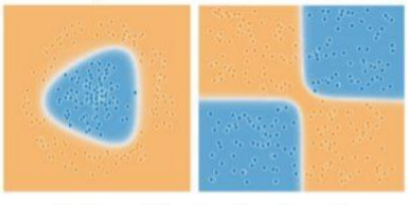
iv. ReLU activation function



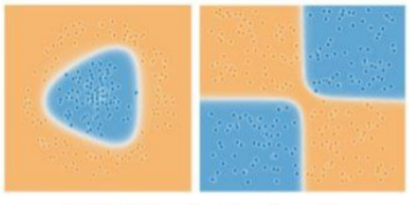
v. Sigmoid activation function



vi. ReLU activation function



vii. Sigmoid activation function



viii. ReLU activation function