# SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

## Department of AI &DS

**Course Name –** 19AD602 DEEP LEARNING

**III Year / VI Semester**

**Unit 3-DIMENSIONALITY REDUCTION**

**Topic:** Training a Convnet: weights initialization,batch normalization

## Case Study: Training a ConvNet with Batch Normalization and Weight Initialization

A team of researchers aimed to optimize a convolutional neural network (ConvNet) for image classification. They experimented with Xavier initialization to prevent gradient issues and used batch normalization layers to stabilize training. This combination accelerated convergence and improved test accuracy by 10%.

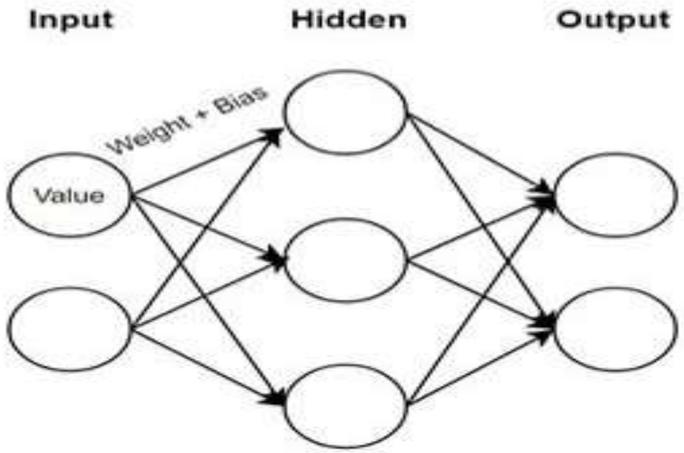
**Activity:**

1. **Dataset Preparation:** Use the CIFAR-10 dataset.
2. **Model Training:**
   - Implement two ConvNets: one with default initialization and another with Xavier/He initialization.
   - Add batch normalization layers to the second network.
3. **Evaluation:** Compare the training speed and accuracy of both networks, and observe the impact of weight initialization and batch normalization.

4o

# Layer Parameters in Neural Networks

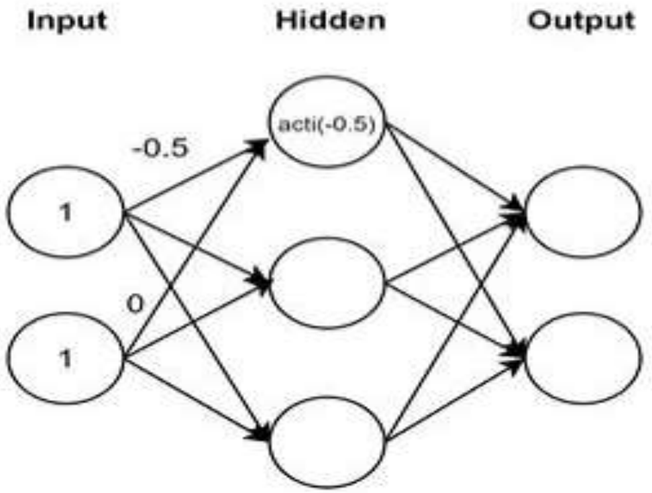
- Weights and Biases
- Layer parameters are learnable and learned during training



# Bias in Neural Networks

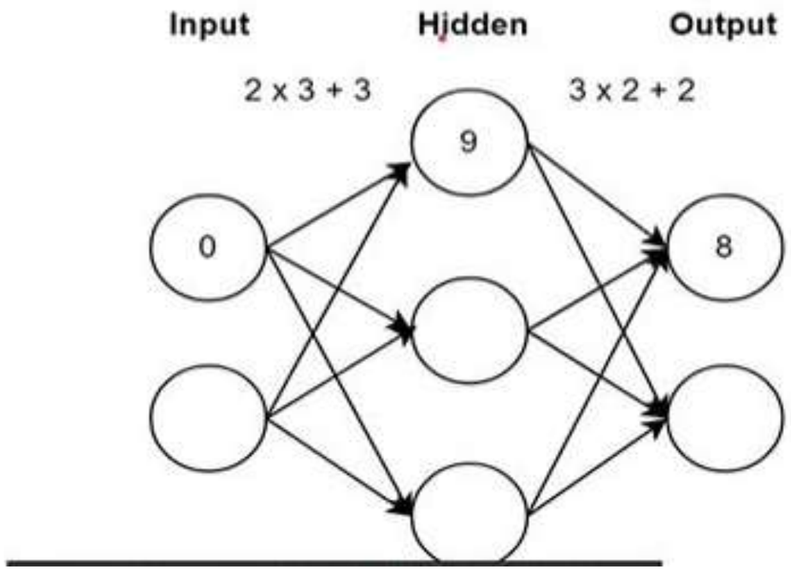
- Determines if a neuron is activated and how much
- Increases the flexibility of the model which is good during training

Output = activation(sum of weights + biases)



# How to find the number of Parameters

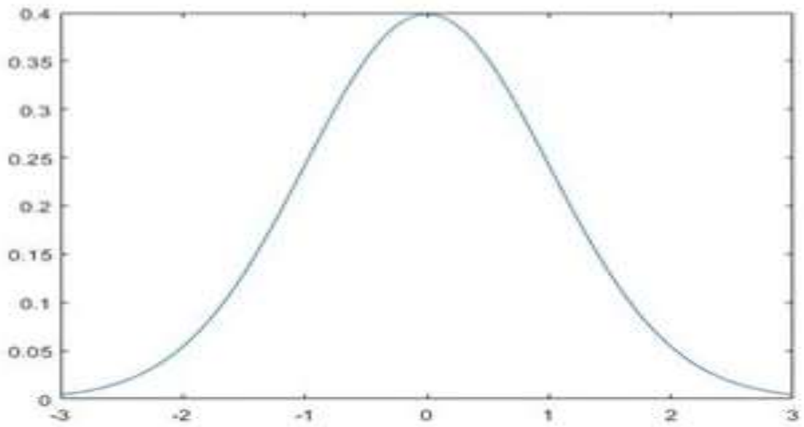
Inputs x Outputs + Biases




```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 4
_________________________________________________________________
dense_1 (Dense)              (None, 24)                72
_________________________________________________________________
dense_2 (Dense)              (None, 12)                300
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 26
=================================================================
Total params: 402
Trainable params: 402
Non-trainable params: 0
_________________________________________________________________
```

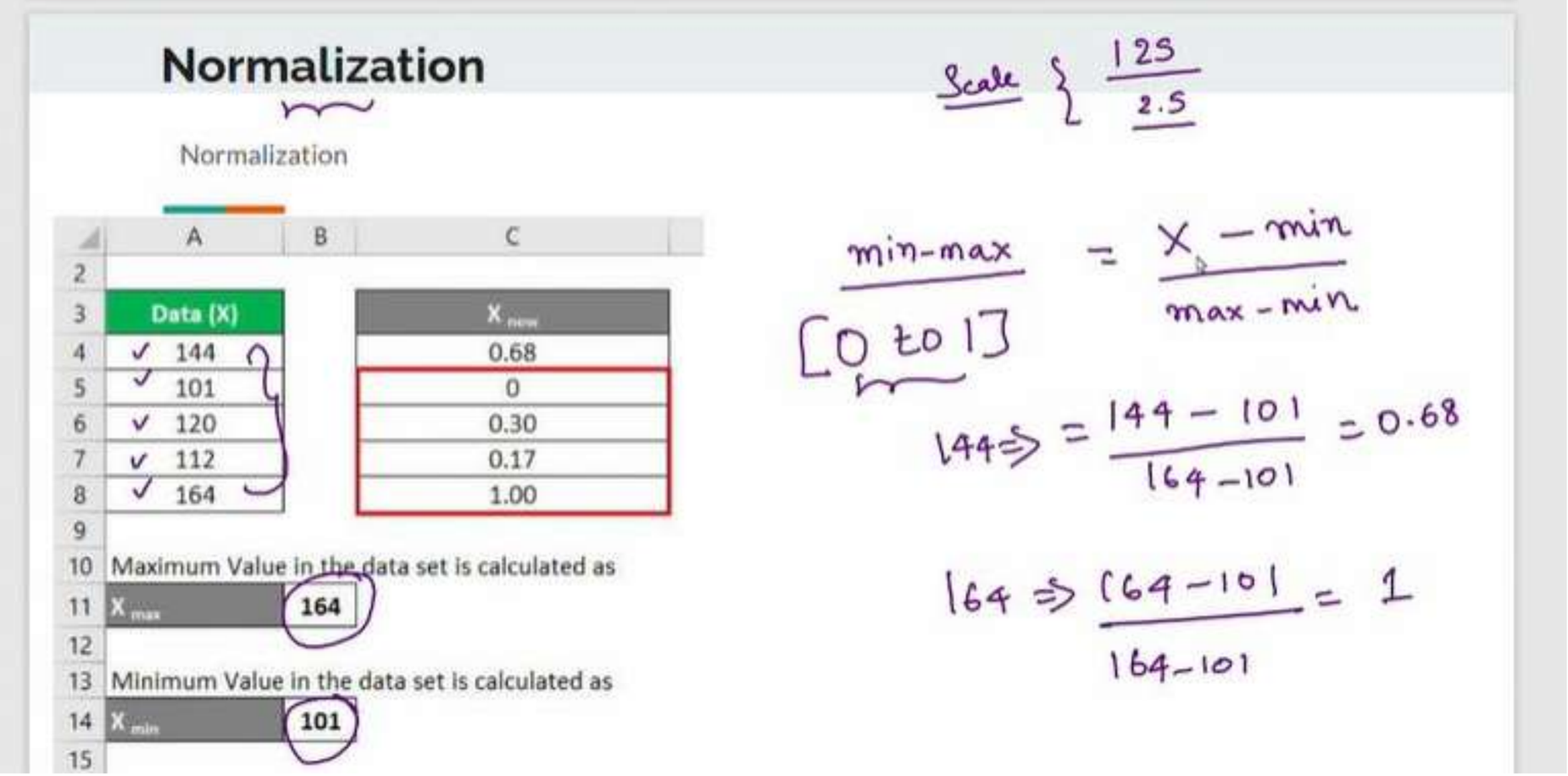
# Weight Initialization

- What is it and why do we use it?
- Use a seed if you want to replicate results



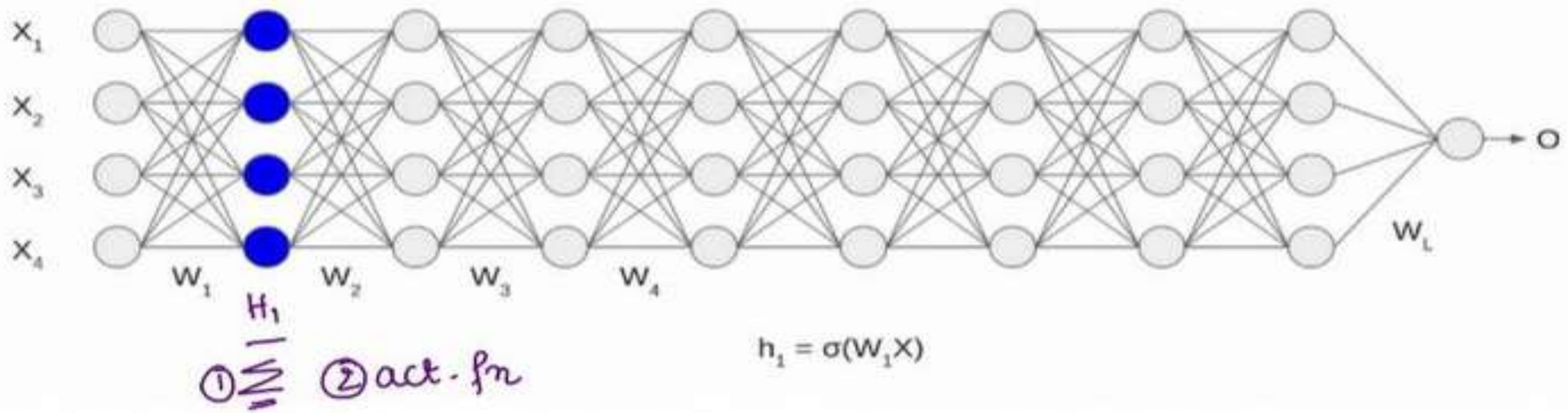

## Layer weight initializers

- RandomNormal class
- RandomUniform class
- TruncatedNormal class
- Zeros class
- Ones class
- GlorotNormal class
- GlorotUniform class
- Identity class
- Orthogonal class
- Constant class
- VarianceScaling class


Normalization
Normalization
A
B
C
Data (X)
$X_{new}$
144
101
120
112
164
0.68
0
0.30
0.17
1.00
Maximum Value in the data set is calculated as
$X_{max}$
164
Minimum Value in the data set is calculated as
$X_{min}$
101
Scale { 125 / 2.5
min-max = $\frac{X - min}{max - min}$
[0 to 1]
144 ⇒ $\frac{144 - 101}{164 - 101}$ = 0.68
164 ⇒ $\frac{164 - 101}{164 - 101}$ = 1

# Training a Convnet: batch normalization

Initially, our inputs X1, X2, X3, X4 are in normalized form as they are coming from the pre-processing stage. When the input passes through the first layer, it transforms, as a sigmoid function applied over the dot product of input X and the weight matrix W.




$H_1$

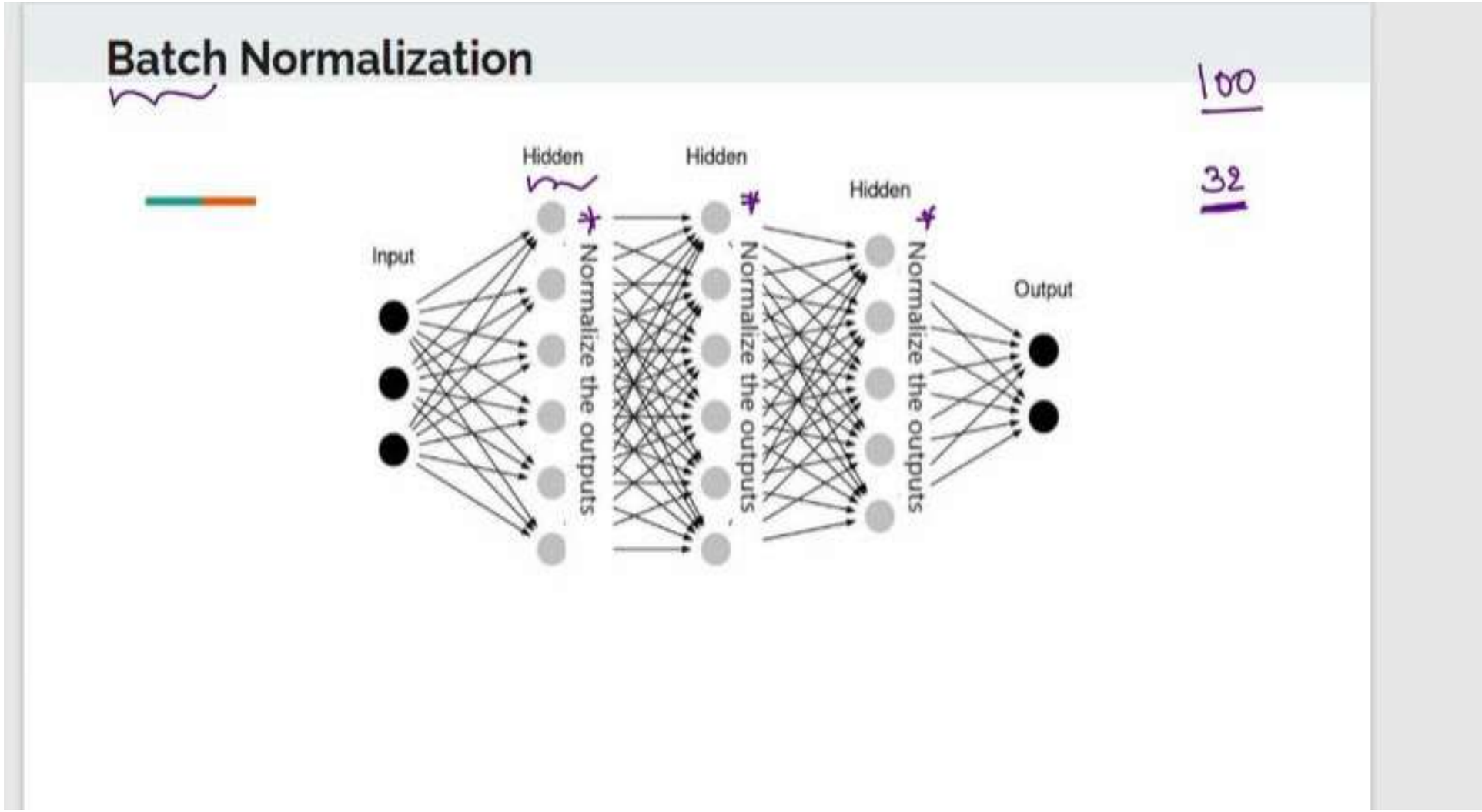
① ≦ ② act. fn

$h_1 = \sigma(W_1 X)$

Similarly, this transformation will take place for the second layer and go till the last layer

* before act fn
* after act fn

# Training a Convnet: batch normalization



# Training a Convnet: batch normalization

1. Normalization is a data pre-processing tool used to bring the numerical data to a common scale without distorting its shape.
2. Generally, when we input the data to a machine or deep learning algorithm we tend to change the values to a balanced scale. The reason we normalize is partly to ensure that our model can generalize appropriately.
3. Now coming back to Batch normalization, it is a process to make neural networks faster and more stable through adding extra layers in a deep neural network. The new layer performs the standardizing and normalizing operations on the input of a layer coming from a previous layer.
4. In a standard feedforward neural network, the batch normalization layer takes the activations of a hidden layer as input, and outputs the normalized and transformed activations, which are then passed to the activation function. During training, the mean and standard deviation of the activations are computed over each mini-batch of data, and the learnable parameters gamma and beta are updated using gradient descent to minimize the loss function of the network.

# Training a Convnet: batch normalization

Here's how batch normalization works:

1. Compute the mean and standard deviation of the activations of each layer in a mini-batch of data.
2. Normalize the activations of each layer by subtracting the mean and dividing by the standard deviation.
3. Scale and shift the normalized activations by learnable parameters, known as gamma and beta, which are updated during training.
4. Pass the normalized and transformed activations to the next layer.

During training, the mean and standard deviation of the activations are computed over each mini-batch of data, and the gamma and beta parameters are updated using gradient descent to minimize the loss function of the network.

We apply a batch normalization layer as follows for a minibatch $\mathcal{B}$:

m → no. of neurons

(1) $$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

(2) $$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$$

(3) $$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$ ← smoothing

(4) $$y_i = \gamma \hat{x}_i + \beta = \mathrm{BN}_{\gamma,\beta}(x_i)$$

Where $\gamma$ and $\beta$ are learnable parameters.

## Example - Mini batch of 4 examples with 1 feature

X = [1, 2, 3, 4]

1. mu = 2.5 # mean of X

   sigma = 1.118 # standard deviation of X

2. X_norm = (X - mu) / sigma

   X_norm = [-1.3416, -0.4472, 0.4472, 1.3416]

3. gamma = 1, beta = 0

   Z = gamma * X_norm + beta

   Z = [-1.3416, -0.4472, 0.4472, 1.3416] pass to next layer

1 , 2 , 3 , 4

Thank you