◾ Batch Normalization:

➢ Batch-Normalization(BN) is an algorithmic method which makes the training of Deep Neural Networks (DNN) faster and more stable.

➢ It consists of normalizing activation vectors from hidden layers using the mean and variance of the current batch.

➢ This normalization step is applied right before (or right after) the nonlinear function.

Covariate shift:

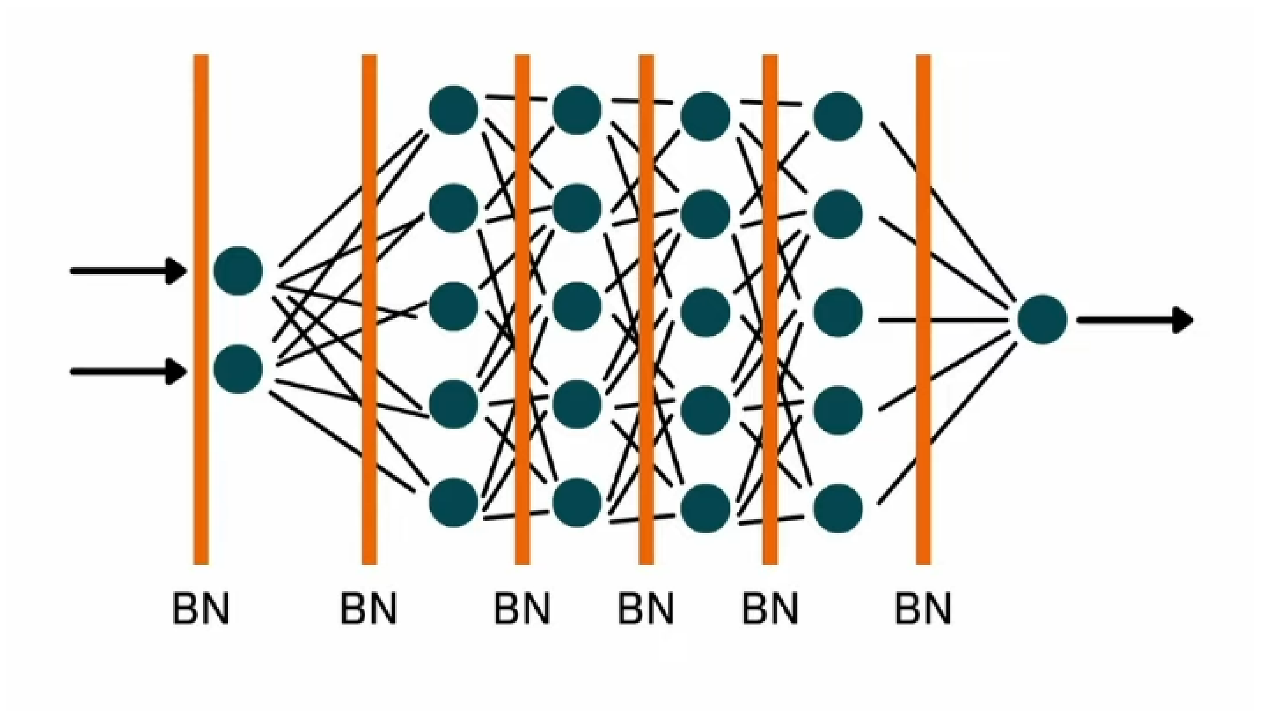
➢ Kunai model train garda input euta distribution diyera train garinxa ra testing ko bela chai chuttai distribution ko data use vako hunxa, jaslai nai covariate shift vaninxa.

Yesto condition maa model retrain garna parxa otherwise model le test data maa ramrari kaam gardaina.

➢Using HU initialization with ELU and other variants of ReLU can reduce the danger of unstable gradients at the beginning of training but at the middle or end of the training gradient problem might rise ,
so **Batch Normalization** might be its solution.

➢ Batch normalization is quite similar to normalization,yesma chai inputs lai normalized garinxa within a range (mean = zero and standard deviation =1) .Hamle simply data lai zero maa center garna khojeko.

➢ Here we center the inputs to zero and normalize the input.



◾ How Batch Normalization (BatchNorm) works :

1) **Normalization**: BatchNorm starts by normalizing the input to a neural network layer. This normalization is done for each feature (or channel in convolutional layers) within a mini-batch of data. The goal is to ensure that the input to the layer has a similar scale and distribution.

- For each feature in the mini-batch, BatchNorm calculates the mean (μ) and standard deviation (σ).
- It then subtracts the mean and divides by the standard deviation to create a normalized feature for each input.

Mathematically, for a feature xx in a mini-batch:

$$x_{normalized} = \frac{x - \mu}{\sigma}$$

2) **Scaling and Shifting**: After normalization, BatchNorm introduces two learnable parameters for each feature: a scaling factor (γ) and a shifting factor (β). These parameters allow the network to adaptively adjust the mean and variance of each feature during training.
   - The scaling factor (γ) determines how much the normalized feature should be scaled.
   - The shifting factor (β) determines how much the normalized feature should be shifted.

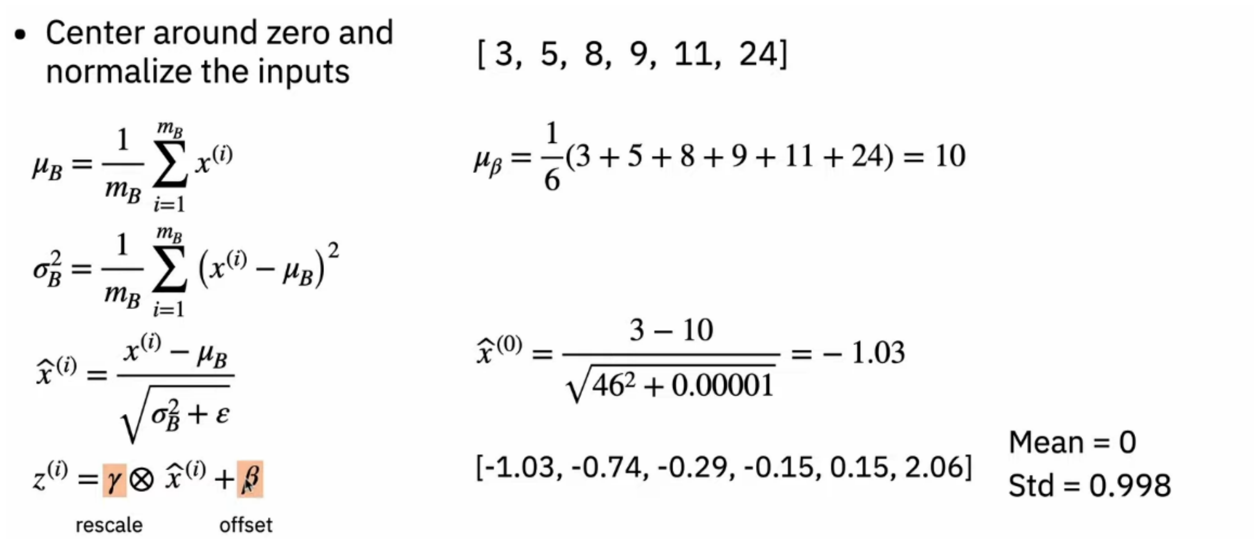
The final output of BatchNorm for a feature is calculated as:

$$y = y * x_{normalized} + \beta$$

This allows the network to learn the optimal mean and variance for each feature during training.

3) **Training and Inference Modes**: During training, BatchNorm computes the mean and variance of each

feature using the mini-batch statistics. However, during inference (when making predictions), it uses a running average of the mean and variance calculated during training on the entire dataset. This ensures that BatchNorm can work effectively on individual examples, even when they aren't part of a mini-batch.



- Center around zero and normalize the inputs

[3, 5, 8, 9, 11, 24]

$$\mu_B = \frac{1}{m_B}\sum_{i=1}^{m_B} x^{(i)}$$

$$\mu_\beta = \frac{1}{6}(3+5+8+9+11+24) = 10$$

$$\sigma_B^2 = \frac{1}{m_B}\sum_{i=1}^{m_B}\left(x^{(i)} - \mu_B\right)^2$$

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

$$\hat{x}^{(0)} = \frac{3 - 10}{\sqrt{46^2 + 0.00001}} = -1.03$$

$$z^{(i)} = \underbrace{\gamma}_{\text{rescale}} \otimes \hat{x}^{(i)} + \underbrace{\beta}_{\text{offset}}$$

[-1.03, -0.74, -0.29, -0.15, 0.15, 2.06]

Mean = 0
Std = 0.998

■ Advantages:-

1. To get rid of gradient problems
2. Achieves same accuracy faster
3. Leads to better performance
4. No need to have standardization layer
5. Reduces the need of regularization
6. Epochs take longer due to the amount of calculations but learn faster.