Batch Normalization

Deep Neural Network

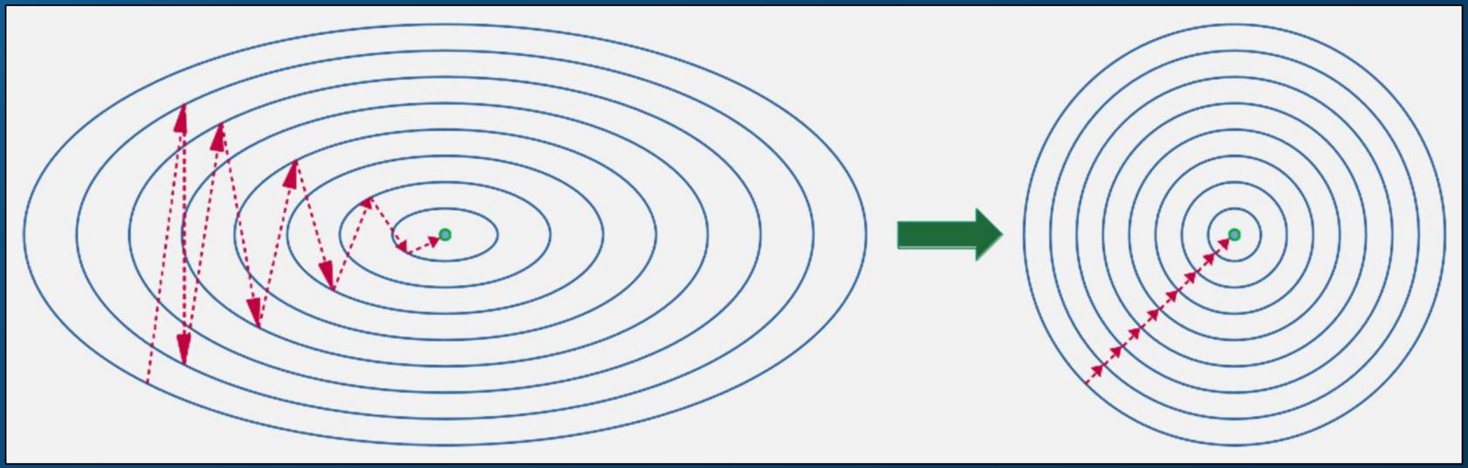
Session 13

Pramod Sharma

pramod.sharma@prasami.com

Batch Normalization

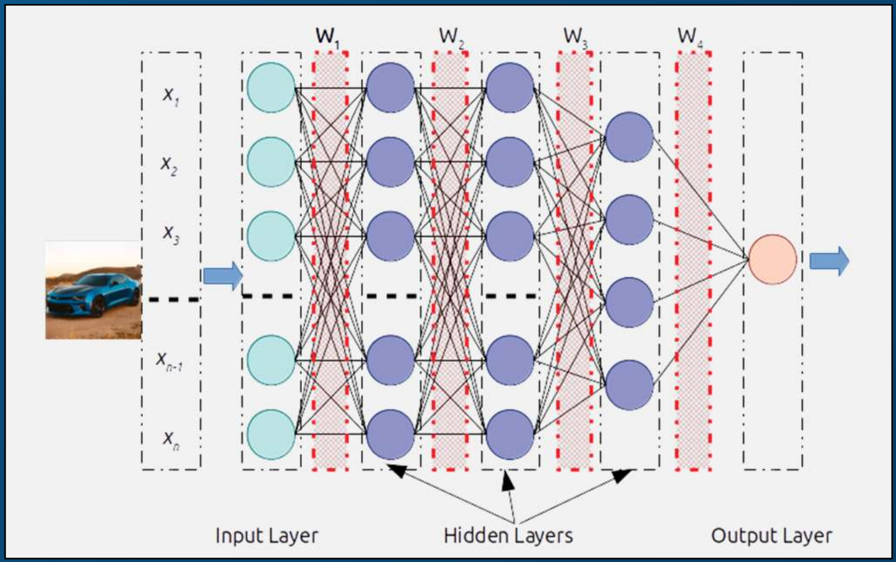
- It definitely helps to normalize input data
- Gradient converges faster



pra-sâmi

Batch Normalization

- What about hidden layer?
- After all activations from previous layer are inputs for current layer…



- Will it help if we normalize the hidden layers too?

Batch Normalization

- Batch normalization (also known as batch norm) [by Sergey Loffe and Christian Szegedy in 2015]
  - Make artificial neural networks faster
  - More stable through normalization of the input layer by re-centering and re-scaling

- Wider choices of hyper- parameter…
  - Higher Learning Rates: Can use much larger learning rates without training diverging
  - Less Sensitive to Weight Initialization: Networks are more robust to poor initializations
  - Reduced Need for Regularization: The inherent noise in batch statistics acts as a regularizer
  - Faster Convergence: All of the above leads to needing fewer epochs to get good results

- In theory, its normalizing activation values of the respective layers

- In practice, it works better if we normalize ‘z’
  - Look at the documentation for details

Batch Normalization

- In General, any Z^i can be normalized

  mean $\mu = \frac{\sum Z^i}{m}$

  std $\sigma^2 = \frac{1}{m}\Sigma(Z^i - \mu)^2$

- $Z^i Norm = \frac{Z^i - \mu}{\sqrt{\sigma^2}}$
- $\hat{z} = \gamma . Z^i$ Norm + β
- where γ *and* β *are paramters we can* train
- if $\gamma = \frac{1}{\sqrt{(\sigma^2)}}$ and $\beta = \frac{\mu}{\sqrt{\sigma^2}}$; Z^i Norm $= \hat{z}$

Batch Normalization

- In General, any Z^i can be normalized

$$\text{mean } \mu = \frac{\sum Z^i}{m}$$

$$\text{std } \sigma^2 = \frac{1}{m}\Sigma(Z^i - \mu)^2$$

$$z^i_{Norm} = \frac{z^i - \mu}{\sqrt{\sigma^2}}$$

$$\hat{z} = \gamma \, . \, z^i_{Norm} + \beta$$

- where γ and β are parameters, we can **Train**
- if $\gamma = \frac{1}{\sqrt{\sigma^2}}$ and $\beta = \frac{\mu}{\sqrt{\sigma^2}}$; $Z^i_{Norm} = \hat{z}$

Instead of using z^i_{Norm}, researchers realized that its better to derive $\hat{z}$ with two trainable parameters.

Intuition is that by normalizing z, we are introducing bias in the system. Hence it makes sense to train these parameters

7

Batch Normalization

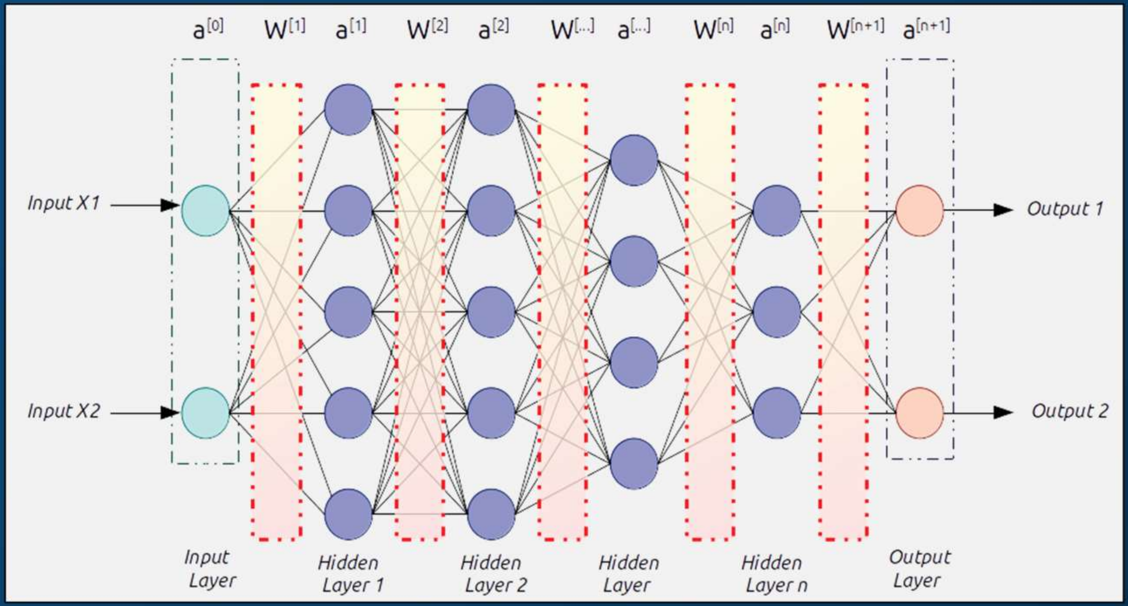
- In General, any Z^i can be normalized
- mean μ = $\frac{\sum Z^i}{m}$
- std $\sigma^2 = \frac{1}{m} \Sigma(Z^i - \mu)^2$
  - Z^iNorm = $\frac{Z^i - \mu}{\sqrt{\sigma^2 + \varepsilon}}$
  - $\hat{z} = \gamma \, . Z^i$ Norm + β
- where γ and β are parameters, we can train
- if $\gamma = \sqrt{\sigma^2 + \varepsilon}$ and $\beta = \mu$; Z^i Norm = $\hat{z}$

Lets add a small ε to prevent zero divide error…

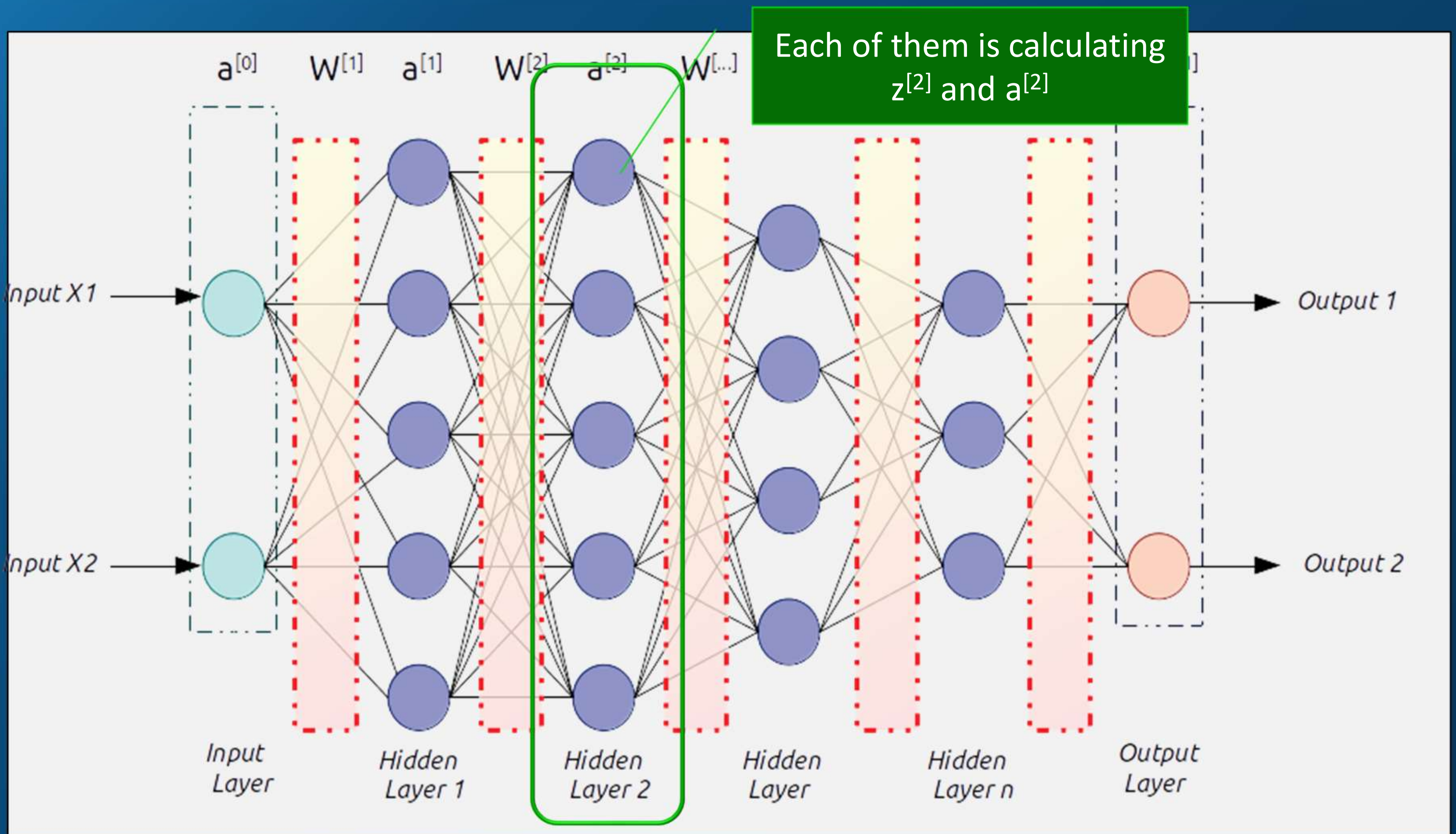
Batch Normalization

- Notes:
  - Batch norm is used along with mini batches
  - Batch norm is applied to the batch under consideration only irrespective of other mini batches

- Where does it fit in overall scheme?

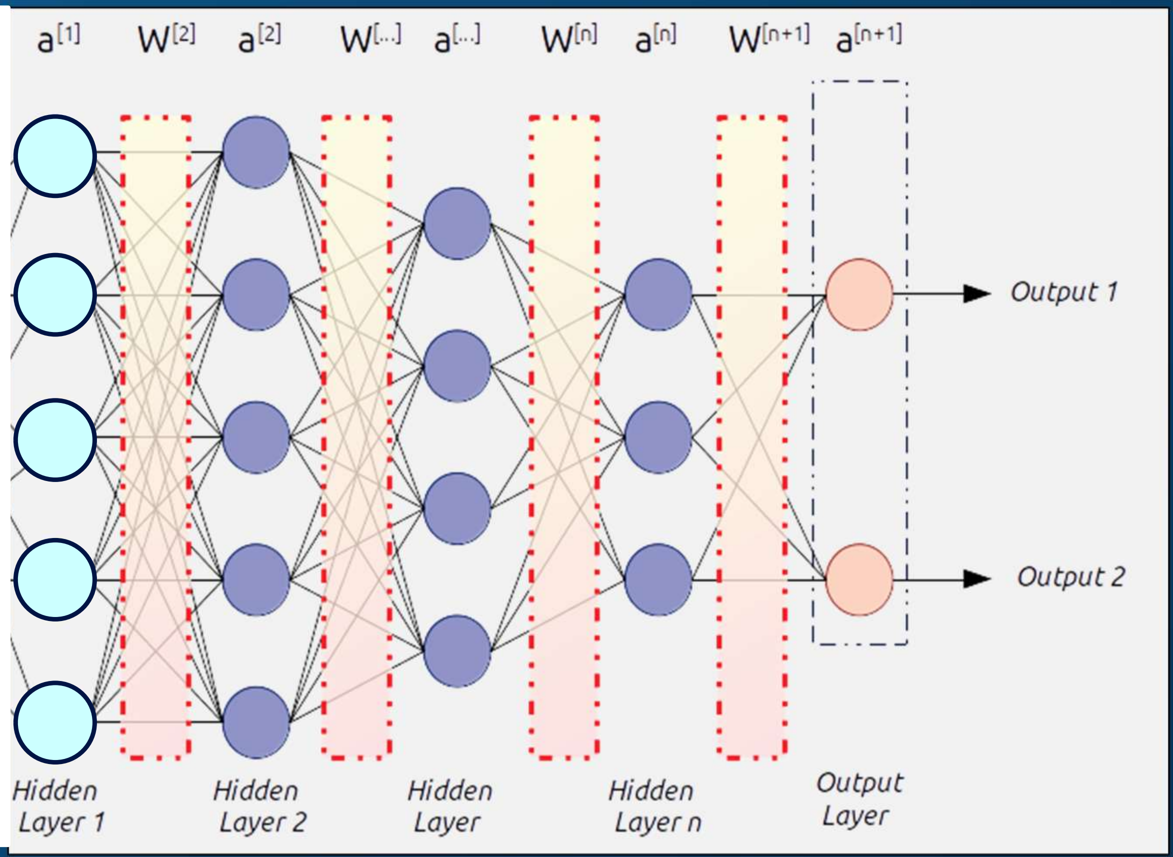


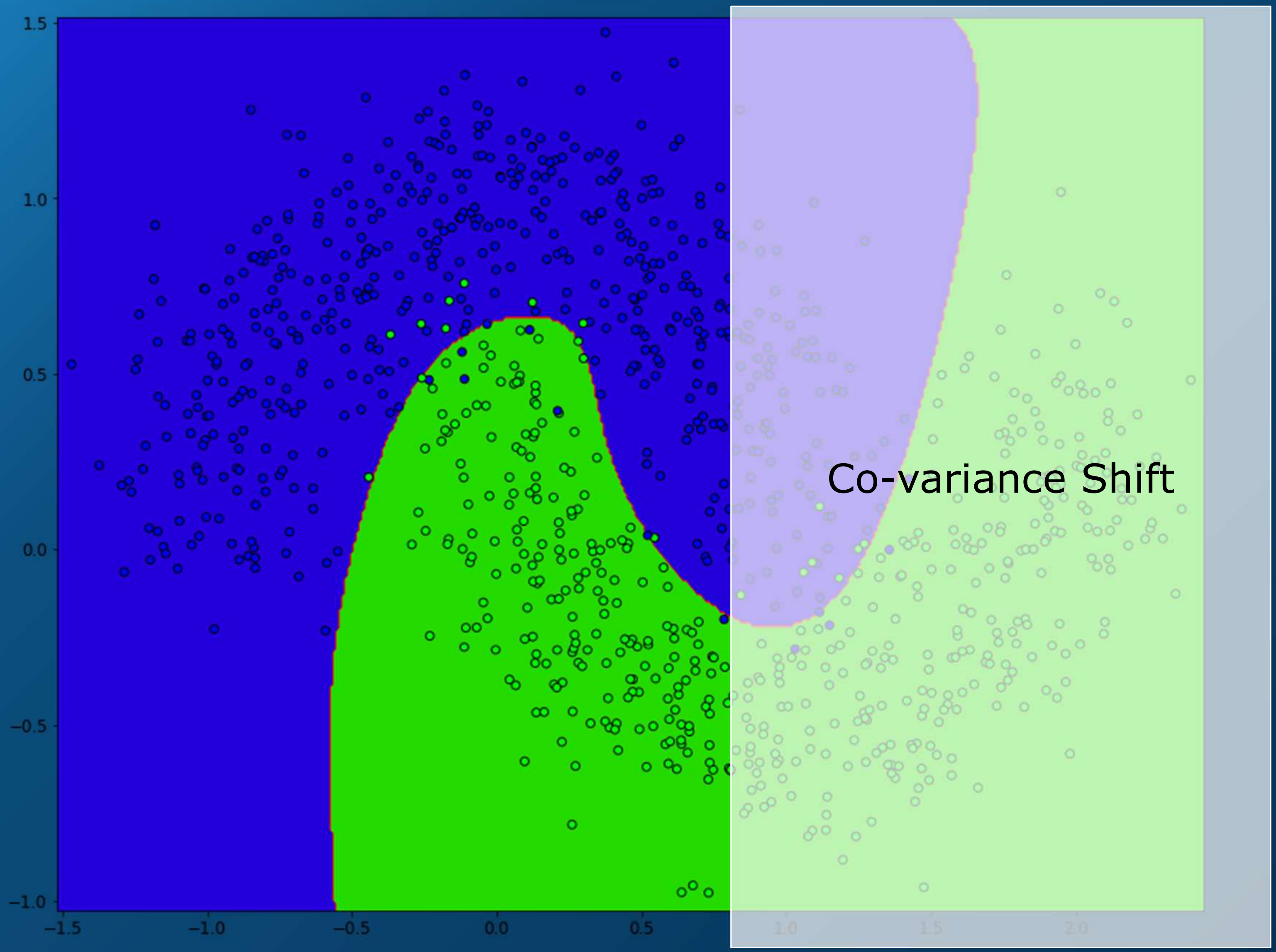
Batch Normalization



Batch Normalization

For $a^{[2]}$ all $a^{[1]}$ are acting as input features

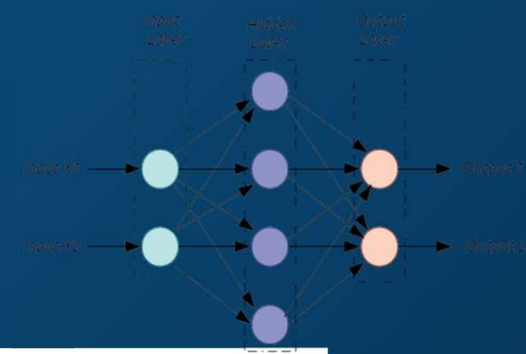




1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Co-variance Shift

Batch Normalization

- Forward and back propagation with batch norm:



W_1, b_1

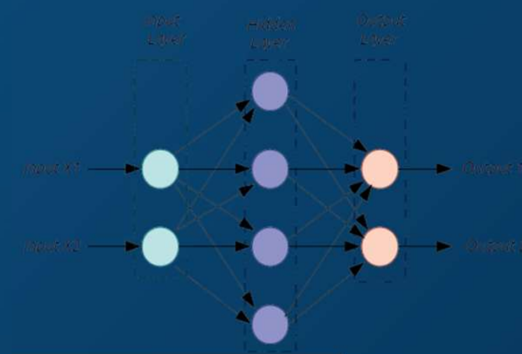
$X \rightarrow Z_1$

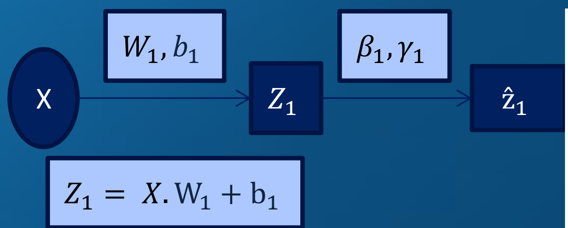
$Z_1 = X.W_1 + b_1$

Our standard equation to calculate z_1.

Batch Normalization

- Forward and back propagation with batch norm:

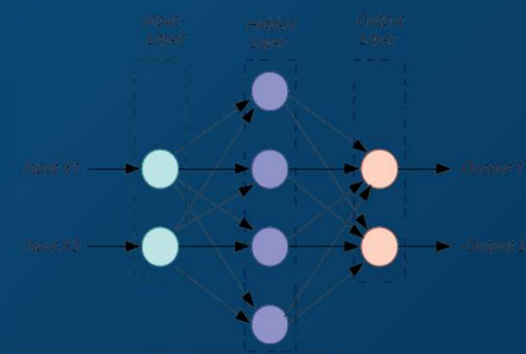




Calculate $\hat{z}_1$, based on β_1, γ_1

Batch Normalization

- Forward and back propagation with batch norm:



X → Z_1 → $\hat{z}_1$ → $a_1 = g_1(\hat{z}_1)$

W_1, b_1

β_1, γ_1

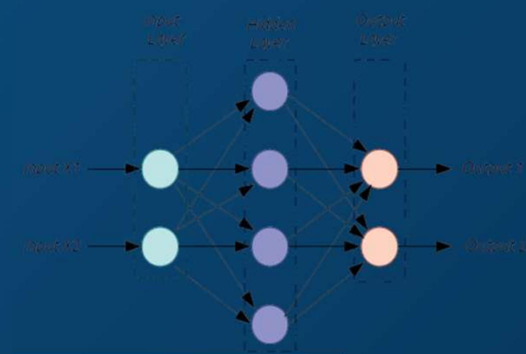
$Z_1 = X.W_1 + b_1$

Apply activation function $g_1(\hat{z}_1)$

pra-sâmi

Batch Normalization

- Forward and back propagation with batch norm:



$$X \xrightarrow{W_1, b_1} Z_1 \xrightarrow{\beta_1, \gamma_1} \hat{z}_1 \rightarrow a_1 = g_1(\hat{z}_1) \xrightarrow{W_2, b_2} z_2$$

$Z_1 = X.W_1 + b_1$

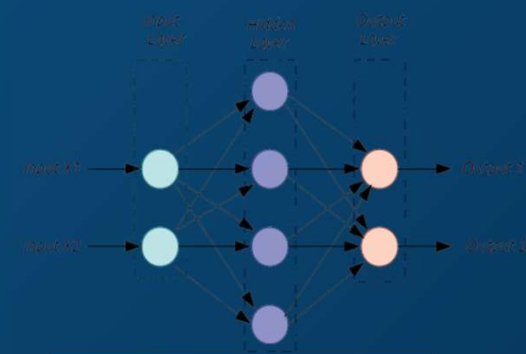
$z_2 = a_1.W_2 + b_2$

Calculate z_2 as usual

pra-sâmi

Batch Normalization

- Forward and back propagation with batch norm:



W_1, b_1

X → Z_1 → $\hat{z}_1$ → $a_1 = g_1(\hat{z}_1)$ → z_2 → $\hat{z}_2$

β_1, γ_1

W_2, b_2

β_2, γ_2

$Z_1 = X.W_1 + b_1$

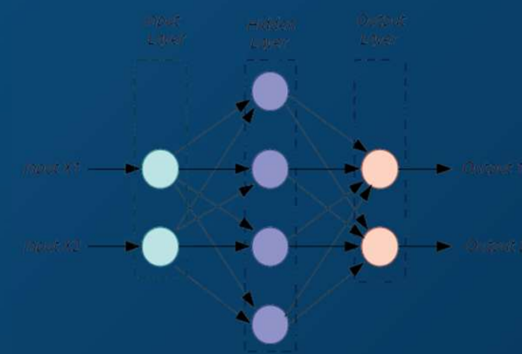
$z_2 = a_1.W_2 + b_2$

We know how to calculate $\hat{z}_2$

pra-sâmi

Batch Normalization

- Forward and back propagation with batch norm:



$X \xrightarrow{W_1, b_1} Z_1 \xrightarrow{\beta_1, \gamma_1} \hat{z}_1 \rightarrow a_1 = g_1(\hat{z}_1) \xrightarrow{W_2, b_2} z_2 \xrightarrow{\beta_2, \gamma_2} \hat{z}_2 \rightarrow a_2 = g_2(\hat{z}_2)$

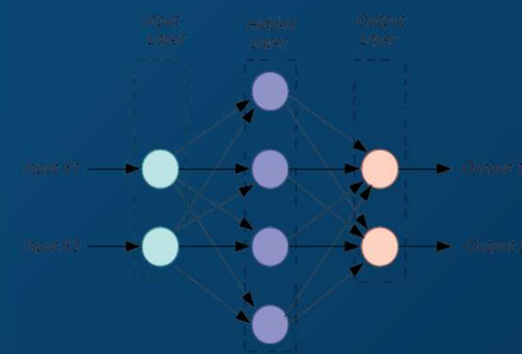
$Z_1 = X.W_1 + b_1$

$z_2 = a_1.W_2 + b_2$

We also know how to calculate a_2

pra-sâmi

Batch Normalization



- Forward and back propagation with batch norm:

$$X \xrightarrow{W_1, b_1} Z_1 \xrightarrow{\beta_1, \gamma_1} \hat{z}_1 \rightarrow a_1 = g_1(\hat{z}_1) \xrightarrow{W_2, b_2} z_2 \xrightarrow{\beta_2, \gamma_2} \hat{z}_2 \rightarrow a_2 = g_2(\hat{z}_2)$$

$Z_1 = X.W_1 + b_1$

$z_2 = a_1.W_2 + b_2$

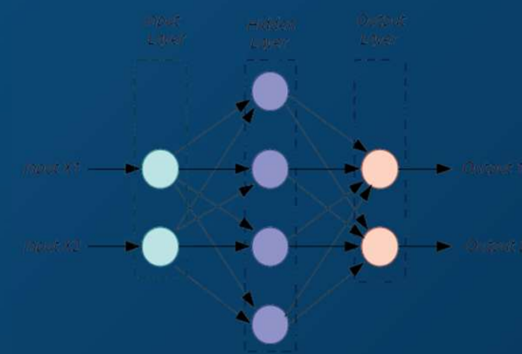
$\beta_1 = \beta_1 - \alpha\,.\,\partial\,\beta_1$
$\gamma_1 = \gamma_1 - \alpha\,.\,\partial\,\gamma_1$

Using the gradient descent, update β's, γ's along with W's and b's

$\beta_2 = \beta_2 - \alpha\,.\,\partial\beta_2$
$\gamma_2 = \gamma_2 - \alpha\,.\,\partial\gamma_2$

Batch Normalization

- Forward and back propagation with batch norm:



$X \rightarrow Z_1 \rightarrow \hat{z}_1 \rightarrow a_1 = g_1(\hat{z}_1) \rightarrow z_2 \rightarrow \hat{z}_2 \rightarrow a_2 = g_2(\hat{z}_2)$

$W_1, \cancel{b_1}$ $\quad \beta_1, \gamma_1$ $\quad W_2, \cancel{b_2}$ $\quad \beta_2, \gamma_2$

$Z_1 = X.W_1 \cancel{+ b_1}$

$z_2 = a_1.W_2 \cancel{+ b_2}$

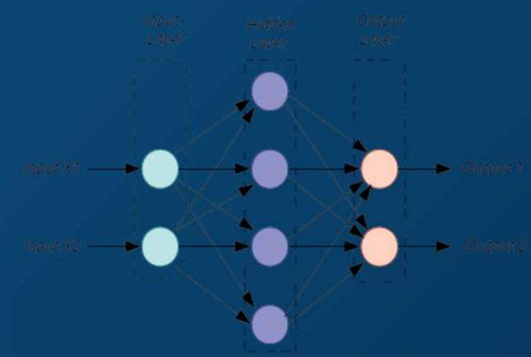
$\beta_1 = \beta_1 - \alpha \,.\, \partial\, \beta_1$
$\gamma_1 = \gamma_1 - \alpha \,.\, \partial\, \gamma_1$

$\beta_2 = \beta_2 - \alpha \,.\, \partial \beta_2$
$\gamma_2 = \gamma_2 - \alpha \,.\, \partial \gamma_2$

One more thing, since we are normalizing our Z's, keeping b's in the equation does not make any sense now. Being the constant it will get eliminated!!

Batch Normalization

- Forward and back propagation with batch norm:



$W_1, \cancel{b_1}$ $\quad$ β_1, γ_1 $\quad$ $W_2, \cancel{b_2}$ $\quad$ β_2, γ_2

$$X \rightarrow Z_1 \rightarrow \hat{z}_1 \rightarrow a_1 = g_1(\hat{z}_1) \rightarrow z_2 \rightarrow \hat{z}_2 \rightarrow a_2 = g_2(\hat{z}_2)$$

$$Z_1 = X.W_1 \cancel{+ b_1}$$

$$z_2 = a_1.W_2 \cancel{+ b_2}$$

$$\beta_1 = \beta_1 - \alpha \,.\, \partial\, \beta_1$$
$$\gamma_1 = \gamma_1 - \alpha \,.\, \partial\, \gamma_1$$

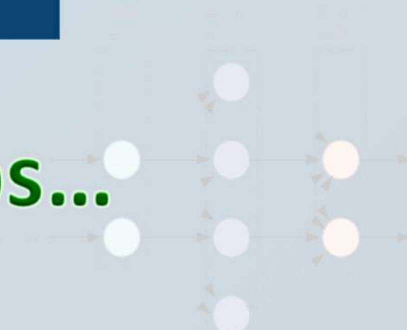
$$\beta_2 = \beta_2 - \alpha \,.\, \partial \beta_2$$
$$\gamma_2 = \gamma_2 - \alpha \,.\, \partial \gamma_2$$

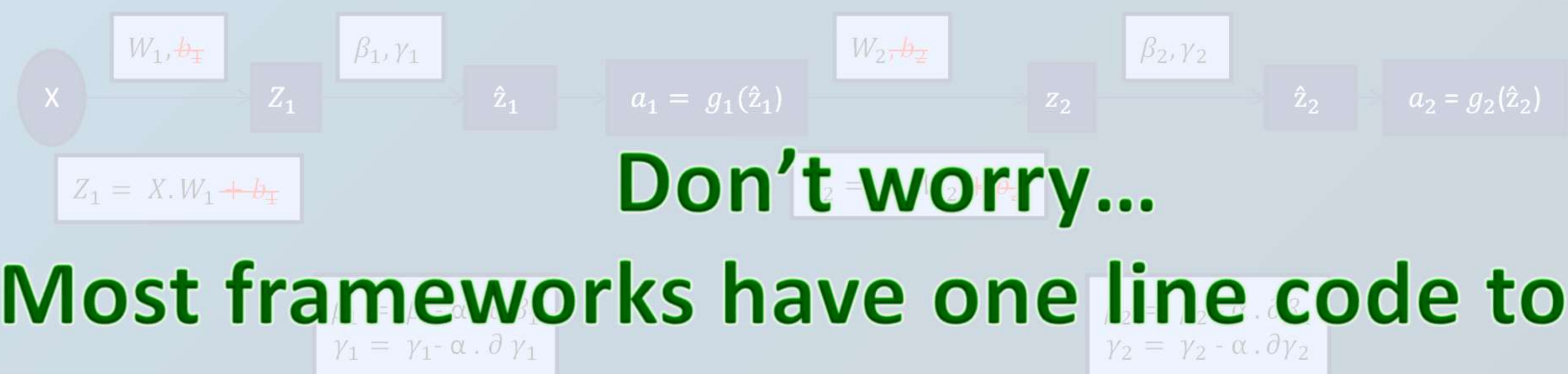
And at test/validation time using a exponentially weighted average!
So while training do not forget to save exponentially weighted values or simply running average!!

Batch Normalization

- Forward and back propagation with batch norm:

Too many calculation steps…





Don't worry…

Most frameworks have one line code to do it.

Batch Normalization – Code Sample

```
model = tf.keras.models.Sequential(
    [
      tf.keras.layers.RNN( keras.layers.LSTMCell(units), input_shape=(None, input_dim)  ),
      tf.keras.layers.BatchNormalization(),
      tf.keras.layers.Dense(output_size),
    ]
  )
```

```
class Net(nn.Module):
        def __init__(self):
        super(Net, self).__init__()
        self.dense1 = nn.Linear(in_features=320, out_features=50)
        self.dense1_bn = nn.BatchNorm1d(50)
        self.dense2 = nn.Linear(50, 10)
```

- And it is applied to mini batches only….
- Batch Norm can be updated using any of the optimization functions…

Batch Normalization

Remember β, γ *are parameters you train*!

At test time, we use these fixed running_mean and running_var values for normalization.

Reflect…

- What is the primary purpose of Batch Normalization in deep learning?
  - A) To prevent overfitting
  - B) To reduce the number of parameters in the model
  - C) To accelerate training and reduce internal covariate shift
  - D) To increase the depth of the neural network
- Answer: C) To accelerate training and reduce internal covariate shift
- At which stage is Batch Normalization applied in a neural network?
  - A) After the input layer
  - B) After the activation function
  - C) Before the loss calculation
  - D) Before or after the activation function, depending on the implementation
- Answer: D) Before or after the activation function, depending on the implementation
- Which of the following is a key step in Batch Normalization?
  - A) Normalizing the gradient updates
  - B) Normalizing the activations by subtracting the batch mean and dividing by the batch standard deviation
  - C) Initializing weights to zero
  - D) Adding noise to the input data
- Answer: B) Normalizing the activations by subtracting the batch mean and dividing by the batch standard deviation
- What are the two learnable parameters introduced in Batch Normalization?
  - A) Gamma and Beta
  - B) Alpha and Beta
  - C) Theta and Gamma
  - D) Sigma and Mu
- Answer: A) Gamma and Beta

Next Session… Recurrent Neural Networks

pra-sâmi