**Day 17**

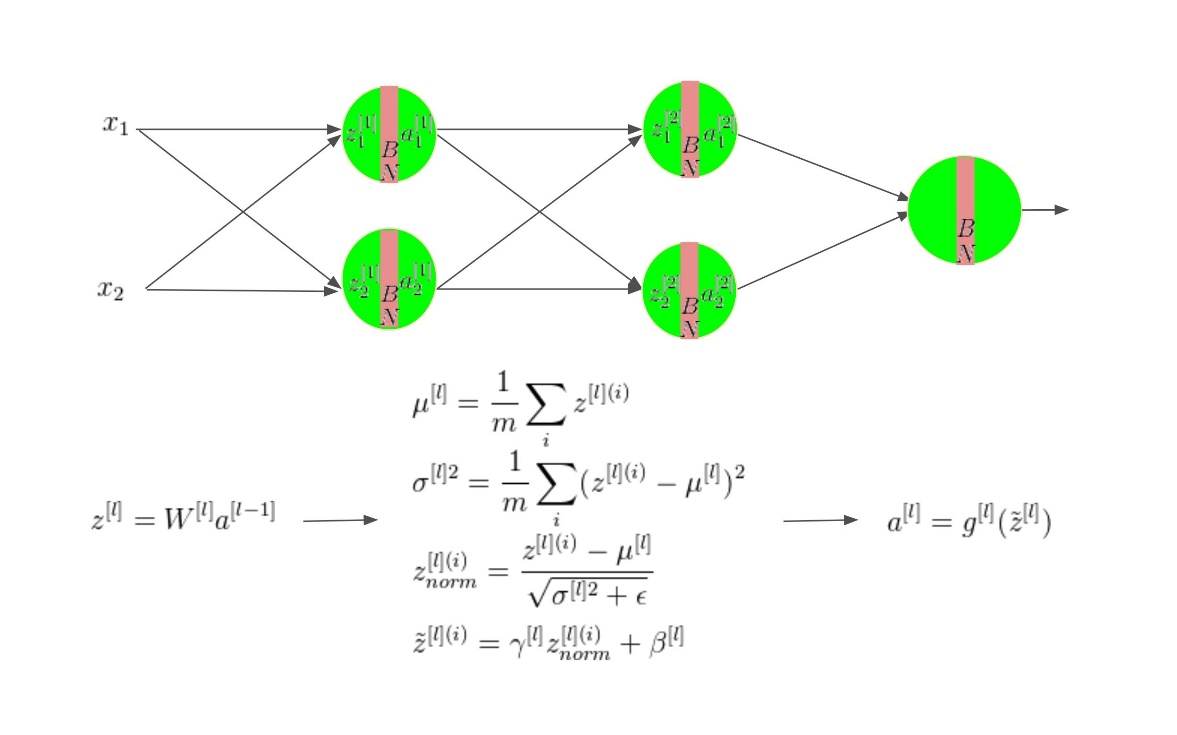
**What to do?**

Learn about Batch normalization

**Batch Normalization:**

We know that most ML models speed up their process when the input features are normalized, using mean and standard deviation. Now, to speed a (deep) neural network, can the same be applied to activation values, say hidden layer 2 activations to train weights and bias of layer 3? We cannot normalize the activations, but we CAN normalize the input that goes into activation function, i.e. (Z[2]).

**Implementing Batch Norm:**



Given some intermediate values in a neural network, Z(1), Z(2),…, Z(m) of a layer l,

(mean) mu = average of Z(i)  - (1)

(variance) var = average of (Z(i) – mu)2 - (2)

Z(i)norm = (Z(i) – mu) / (var + epsilon) - (3)

Z͂(i) = γ \* Z(i)norm + β - (4)

Here, γ and β are learning parameters. Hence, they need to be updated using any optimizer

Once the intermediate values are normalized, they are then inserted in activation function. The process is iterated until the output layer.

This technique is usually applied with mini – batch gradient descent, yet not limited to. However, if batch normalization is applied to mini – batch gradient, then the equations 1 and 2 are performed at every mini – batch.

**Why does it work?**

Batch normalization works because its process is analogous to input features normalization, except in this case, the hidden units’ intermediate values are being normalized. This helps layers robust to changes and has a very slight effect of regularization.

**Python:**

Fashion MNIST dataset that underwent two models (with and without batch normalization).

