**BATCH NORMALIZATION**

**Introduction**

Batch normalization, on a high level is a technique that improves the training of a neural network by stabilizing the input distribution between the hidden layers to which it is applied to. The implementation of the technique involves adding additional network layers that controls the mean and variance of these input distributions.

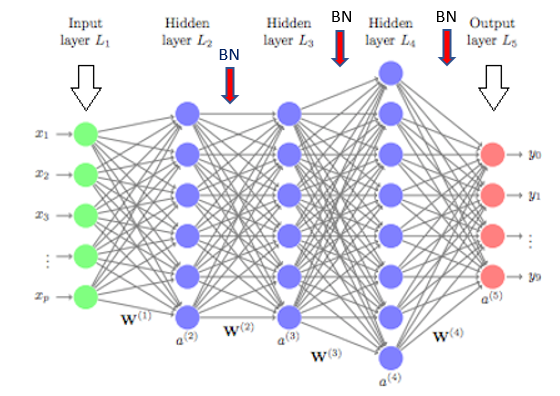
**Why Batch Normalization?**

In general, Normalization is applied to the input feature data in order to represent all the feature values across a single scale from 0 to 1. This is done to eliminate the impact or domination of one feature over another in the prediction process. Another reason with respect to neural networks is that, the forward propagation involves the calculation of dot products of the weights and features values, which might result in very high output values, high computational times and increased memory usage, with unnormalized data. A similar issue occurs with back propagation, during gradient descent process. Hence, the input data is normalized to train the network faster and accurately.

The above approach works, if we have a single activation layer. However, when a neural network has additional hidden layers, normalizing the input data alone might still not be enough, as the distribution of the data passing through these hidden layers keeps changing during the training process, as the weights and the bias of the previous layers change. This phenomenon is referred as Internal covariant shift. This in turn increases the training time and makes it very difficult to train the model. Batch normalization addresses this problem, by normalizing the input data between the hidden layers in the network.

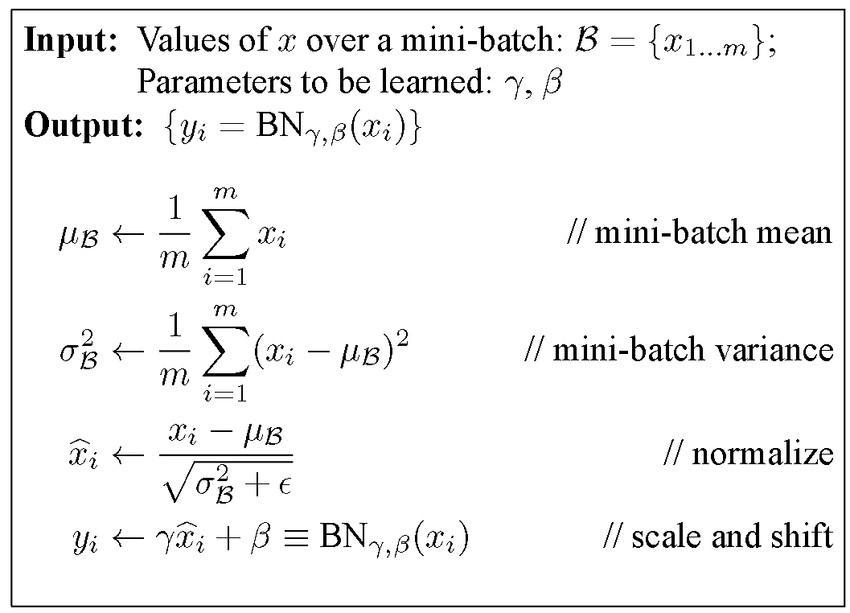
**How Batch Normalization works?**

Considering a neural network as shown below in Figure 1, the Batch normalization layers can be added where the red arrows are pointing, to handle the data passed from layer2, layer3 and layer4. The Internal Covariant shift more specifically refers to the change in the distribution of the activation values of each neuron associated with the layers, due to the varying weights and biases during training. As a result of this, all the hidden activation units passed from layer2 to layer3 and layer3 to layer4 will be normalized. Implementing this transformation for the entire training set at a time might not be practical for a very large dataset and will slow down the training process. Hence batch normalization is applied in mini batches of the training set and the estimated mean and variance of this mini batch will be used for the normalizing the data.



Figure

Although normalization means transforming the mean and variance to 0 and 1 respectively for the data, just doing that can reduce the expressive power of each neuron in the layer and can change what each activation unit represents. In order to make the transformation meaningful and better than the original distribution, two additional parameters **gamma** and **beta** are introduced for each activation unit. These two values will be used to scale and the shift the normalized value and allows the data to have a new mean and variance so that the representation of the original activations is restored. The mean, variance, gamma and beta are trainable parameters and will be learnt during the training process. As a result, these parameters will also be involved in the gradient calculation during backpropagation. The figure below explains the transformation that will be applied to each activation unit in the layer during the forward propagation.



Figure

**Benefits of Batch Normalization**

As a result of applying batch normalization, the network will have normalized data coming in and normalized data within the model thus reducing the training time and eliminating the imbalance with the input distributions. The back propagation using the trainable parameters ensures that the model continues to learn the input distributions exhibiting internal covariant shift and thereby stabilizes the training process. Batch normalization also allows us to use higher learning rates thus accelerating the training process for a faster convergence. In some cases, it also acts as a regularizer and enables us to create a better generalized model even without a Dropout.

**References**

1. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
2. <https://stackoverflow.com/questions/4674623/why-do-we-have-to-normalize-the-input-for-an-artificial-neural-network>
3. <https://orbograph.com/deep-learning-how-will-it-change-healthcare/>
4. Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. In *Advances in Neural Information Processing Systems* (pp. 7694-7705).