# Batch Normalization - Paper Review

Yamina Hamidi

Jully 2021

Paper : IOFFE, Sergey et SZEGEDY, Christian. Batch normalization : Accelerating deep network training by reducing internal covariate shift. In : International conference on machine learning. PMLR, 2015. p. 448-456.

## 1 Motivation

Data is traditionally only scaled and/or normalized, as part of the data pre-processing step, before input to the first layer in a Neural Network. It's well known that relatively large weights can cascade through the network and cause the gradient to vanish. Large data values can cause instabilities in the network. The larger the value the more likely it is that the sigmoid's value is gonna be larger and that it's derivative is gonna tend to zero.

To elaviate this problem the paper suggests to scale each mini-batch of data before input into the hidden layers, which is believed to keep the distribution of the data stable throughout the training process.

## 2 Algorithm

Let $W$ be the input data. To normalize each mini-batch, the mean $\mu_B$ and standard deviation $\sigma_B$ of the mini-batch $B$ are computed :

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} w_i$$

$$\sigma_B = \frac{1}{m}\sum_{i=1}^{m}(w_i - \mu_B)^2$$

The scaled/normalized input batch $W_{norm}$ is given by :

$$W_{norm} = \frac{W - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

The scaled and shifted batch $Z$ (ready to be input to the next layer) is given by :

$$Z = \gamma W + \beta$$

$\epsilon$ is a constant for numerical stability and $\gamma$ and $\beta$ are trainable parameters.

In the training phase the mean and variance of each mini-batch are computed, used to normalize the said batch and update the average mean (moving average) and variance (moving variance) needed to normalize the data in the inference phase. The computation of the moving statistics is done following the formula of the exponential moving average [Wikipedia] :

$$\mu_{moving} = t \quad \mu_{moving} + (1 - t)\mu$$

$$\sigma_{moving} = t \quad \sigma_{moving} + (1 - t)\sigma$$

where $t$ is a constant between 0 and 1.

# 3 Application

To asses the impact that Batch Normalisation has on network performance it is used, in this section, in the well-known LeNet on the MNIST digits database.

## 3.1 LeNet

LeNet is a convolutional network first introduced in Le Cun et al.(1998). It's architecture is composed of 3 convolutional layers and 2 fully connected layers.

Implementation is done in the PyTorch framework. The $32 \times 32$ input shape is kept even if images are $28 \times 28$ in the MNIST database (it is said in the article that it enhances the performance) even if modern implementations don't use it. Also keeping the tanh (the one used in the article) as activation function even though in modern implementation ReLU is more commonly used.

## 3.2 Performance of LeNet

The performance of LeNet on the MNIST database was evaluated using the loss and accuracy.

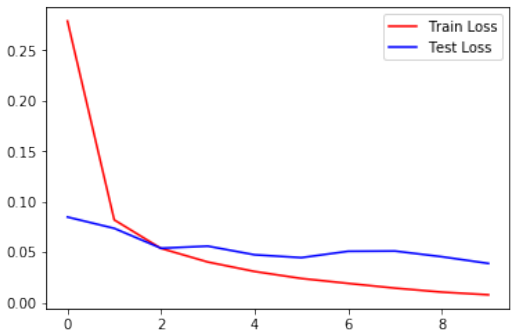


FIGURE 1 – Training and Test Loss of LeNet over 10 epochs. Learning rate = 0.9. Optimizer : SGD. Loss : Cross Entropy.

Figure 1 presents the training and test loss obtained with LeNet. The overall total test accuracy of the model is 98%.

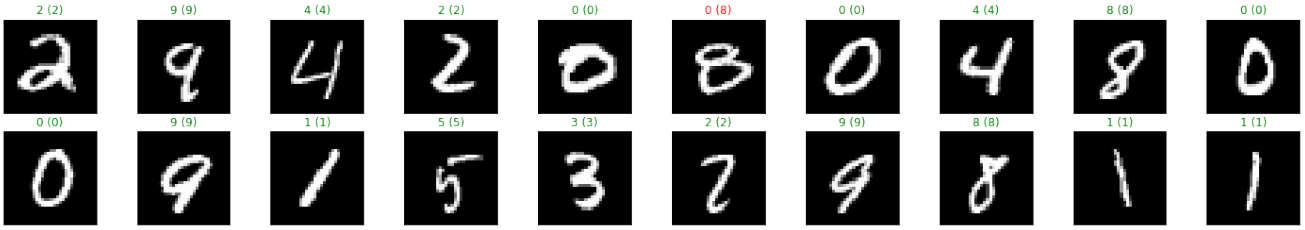


FIGURE **2** – Test samples. The predicted and true label appear on top of the images in green when they match and red when they don't.

Figure 2 shows samples from the test dataset. This sample is representative of the whole database as the model is able to accuratly predict the label of 98% of the samples. Lower per-class accuracy values are observed for the "10" and "8" classes.

### 3.3 Performance of LeNet with Batch Normalization

Batch Normalization is also implemented in PyTorch and it is applied to the output of the 3 convolutional layers and the first fully connected. Comparisons between the BatchNormalization module of PyTorch and the one implemented were carried out and no significant difference in running time and accuracy was noticed.

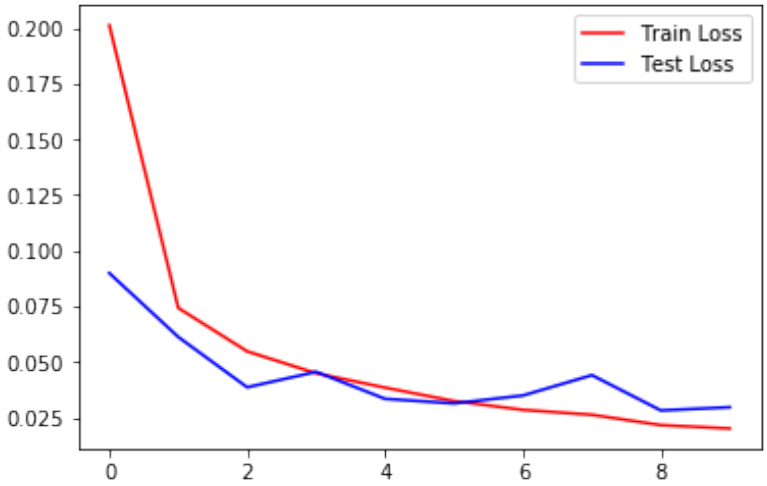


FIGURE **3** – Training and test Loss of LeNet with Batch Normalization over 10 epochs. Learning rate = 0.9. Optimizer : SGD. Loss : Cross Entropy.

Figure 3 presents the training and test loss obtained with LeNet with Batch Normalization. The overall total test accuracy of the model is 98%.

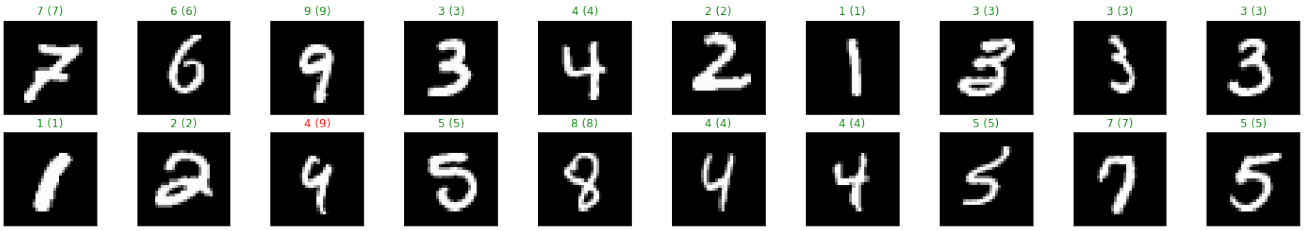


FIGURE 4 – Test samples. The predicted and true label appear on top of the images in green when they match and red when they don't.

Figure 4 shows samples from the test dataset. This sample is representative of the whole database as the model is able to accuratly predict the label of 98% of the samples.

# 4 Conclusion

Batch Normalization can allow for larger learning rates, decrease training time, decrease the amount of regularization needed and remove the necessity of drop-out layer. The above mentionned advantages were not noticed when using the LeNet model generated with batch normalization on the MNIST digits database compared to the model generated without batch normalization. This is probably du to the fact that this model is relatively shallow, not as complexe as Inception the one used in IOFFE and SZEGEDY (2015).