Simplifying Batch Normalization with Removal of Normalization Scale Parameter

1 Abstract

We are testing a modification of the Batch Normalization (BN) technique in deep neural networks. BN methods typically utilize learned scale and shift parameters (gamma and beta) for normalization. We seek to simplify the algorithm by removing the gamma parameter and incorporating a weighted average between consecutive mini-batches' standard deviations. The resulting accuracies show insubstantial differences between the orginal BN algorithm and the simplified one, suggesting that it may not be necessary to learn extra parameters over the course of training.

2 Introduction

Batch Normalization is a method created by Ioffe and Szegedy (2015). BN normalizes the input of each layer for each mini-batch, helping to improve training speed and stability. This is achieved by adjusting the activations of a layer to have a mean of zero and a standard deviation of one, followed by a scale and shift transformation defined by the parameters γ and β. The original BN formula is defined as:

$$BN(x) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}_{\mathscr{B}}}{\hat{\sigma}_{\mathscr{B}}} + \beta. \tag{1}$$

Where x refers to the input from the previous layer, $\hat{\mu}_{\mathscr{B}}$ refers to the mini-batch mean, and $\hat{\sigma}_{\mathscr{B}}$ refers to the mini-batch standard deviation. γ and β are learned, elementwise scale and shift parameters, respectively, that help recover certain amounts of variation from the previous layer.

Particularly, γ is an essential learned parameter as it controls the scaling factor of the normalization of the activation. Due to γ being learned in training, it has the potential to revert the normalization back to the original scale if it is deemed optimal to do so. This makes γ an interesting parameter to study, as it potentially interacts with the mini-batch's standard deviation and may be an incredibly important parameter in the stabilizing effect of BN.

In this study, we try a modification to the original BN algorithm by removing the gamma parameter. Instead, we introduce a weighted average of the standard deviations from the current and previous mini-batches. This approach aims to maintain the stabilizing benefits of BN while potentially reducing its computational complexity and improving the network's ability to generalize. As weighted averages need some way to determine the weighting, we introduce a hyperparameter α that represents the weight of the standard deviation for the previous layer's standard deviation, or the moving standard deviation. The new proposed formula is as follows:

$$BN^*(x) = \frac{\mathbf{x} - \hat{\mu}_{\mathscr{B}}}{\alpha * \hat{\sigma}_{\mathscr{B}-1} + (1-\alpha) * \hat{\sigma}_{\mathscr{B}}} + \beta. \tag{2}$$

We hypothesize that the modification will perform just as good as the original BN algorithm. We will also investigate the effects of different weighting for the two layers to see the importance of the previous activation's scale in training the model.

3 Data and Approaches

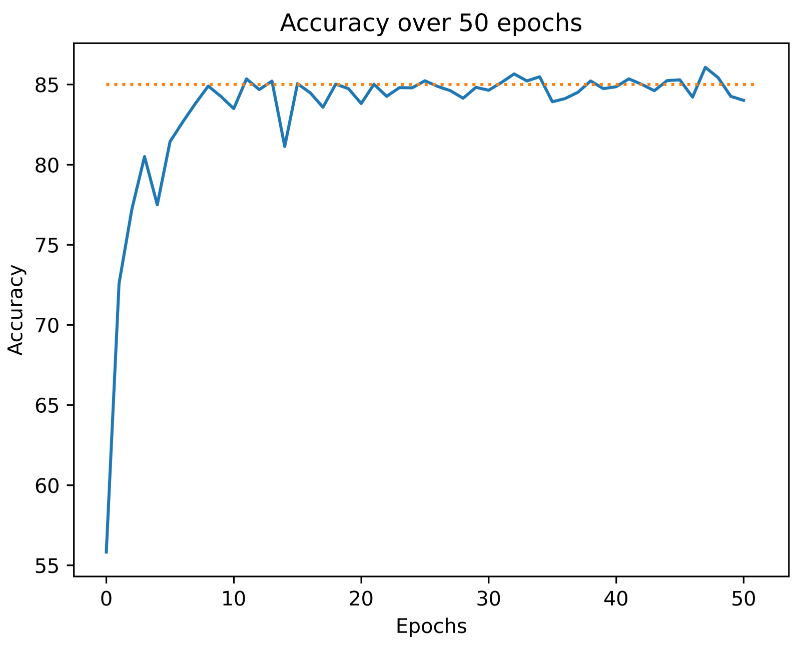
The data used for this study is the CIFAR-10 dataset. The network's architecture is a deep convolutional neural network. Three networks were constructed - one network without the use of BN, one utilizing the original BN algorithm, and another with the modified formula. All networks consist of 6 total convolutional layers separated into three stages, each consisting of 2 convolutions that are followed by a batch normalization, then a ReLU activation. Each stage is separated by max-pooling layer. After the convolutions and pooling, the network transitions into the fully connected layers, reducing dimensionality to 10 to represent the 10 potential labels of the image in the CIFAR dataset. The modification for the BN algorithm was created in a custom pytorch module, and models utilizing the modified algorithm were initialized with a unique α hyperparameter to control weighting.

Our training and testing batch sizes for the models were 64 and 1000, respectively. Training was conducted over a maximum of 50 epochs or until accuracy reached a threshold of 90%. Each epoch's accuracy was stored and compared in a graph to track patterns between the control network and the experimental networks. Models were trained with the ADAM optimizer at a learning rate of 0.001.

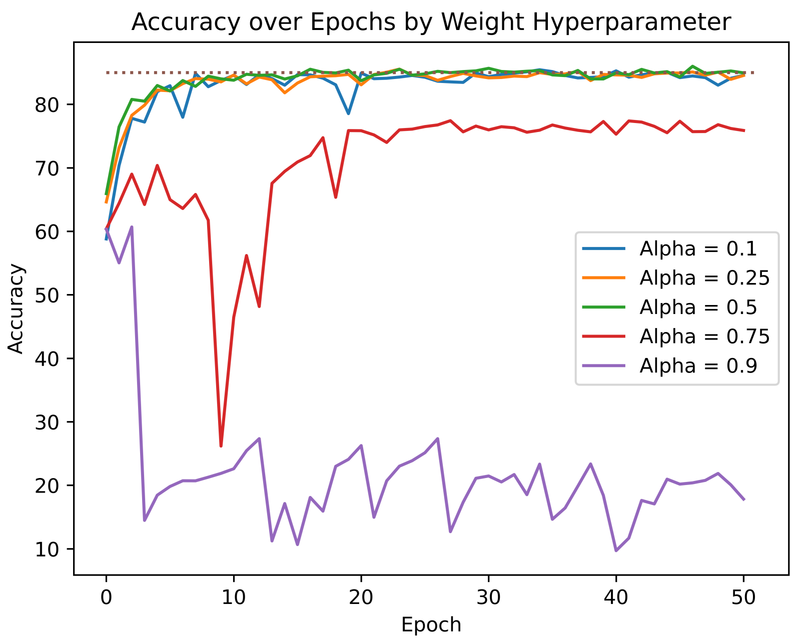
5 unique α levels were tested to study the effects of the importance between the previous and current layer's standard deviations: 0.1, 0.25, 0.5, 0.75, and 0.9.

4 Results and Discussion

The baseline assumption is that Batch Normalization improves accuracy and convergence. This is verified by our BN-less network, which converged on an accuracy of around 75%.



Pictured above, the control benchmark of the network with standard Batch Normalization shows a convergence around the 85% accuracy mark after epoch 10. After 50 epochs of training, the max accuracy attained was 86.4%. Thus, for our modified BN algorithm, 85% was used as the benchmark to indicate that the new network converged/trained properly.



The networks with modified BN were also trained according to the above guidelines, with the maximum accuracy level achieved by any network being 85.44%. Noticeably, α levels of 0.5 and below achieved the 85% benchmark defined by the standard model. The model with α of 0.75 converged at around a 75% accuracy after experiencing large dips in accuracy in the earlier epochs (when most models began converging). The model with α of 0.9 failed to converge after 50 epochs, having a final accuracy of 17.8%.

The results of the test prove promising. The removal of γ from the Batch Normalization algorithm did not have a substantial effect on the accuracy of the resulting models for those with $\alpha <= 0.5$, meaning that a weighted average that at least puts more weight on the current layer's variance performs just as well as models with standard Batch Normalization. Conversely, models that tried to weight the moving variance more performed much worse, in one case not converging at all.

One possible explanation for this is that a weight that emphasizes the current batch's standard deviation provides a smootherbatch normalization, as the value of the resulting weighted standard deviation is more applicable to the current batch. This way, the resulting inputs are more likely to be truly normalized. Another reason may be that the network was not deep enough to expose the modified algorithm to enough periods of normalization, thus biasing our results. In this case, one can imagine how having a learnable normalization scale parameter would help, as each Batch Normalization would have the capability to prevent unwanted, accumulated spillovers from previous layers as our simplification does now.

5 Conclusion

The removal of the γ parameter and replacement with a weighted standard deviation average in the Batch Normalization algorithm had promising results, with three of the five tested models (models with α levels below 0.5) achieving a similar convergence and accuracy to the model with the standard BN at 85%. This suggests that the γ parameter is not necessary to learn during training for proper convergence or high accuracy in a given architecture. However, further testing on deeper models, not necessarily convolutional networks, needs to be conducted on deeper models to truly understand how internal covariate shift occurs in order to verify this simplification.