Investigating the Effect of Batch Normalization on the Hessian of a Two-Layer Neural Network

Mia Reyes
MAE Department, UCLA
miareyes10@ucla.edu

Helene Levy
MAE Department, UCLA
hjlevy@ucla.edu

Abstract

The difficulty of training deep neural networks has led to a proliferation of optimization algorithms which seek to simplify learning and improve convergence. Batch normalization is one of the most well known, and implemented methods to optimize deep neural networks. This study seeks to understand why batch normalization leads to faster convergence through experimental results and analyses on the loss function's Hessian. Two shallow networks with and without batch normalization were trained on the MNIST Dataset and evaluated. We hypothesize that batch normalization leads to faster convergence due to the improvement of the Hessian's condition number, and we support this hypothesis with the demonstration of a reduced pseudo-condition number and improved convergence when batch normalization is introduced.

1. Introduction

Deep learning is a machine learning technique that has gained popularity in recent decades for its success in solving such tasks as image classification and speech recognition [10]. This success is due in large part to several key advancements in the architecture of deep neural networks. One particularly useful development was the introduction of batch normalization, a method of normalizing hidden layers' inputs that has been shown empirically to increase convergence speed [3].

Although batch normalization has demonstrable success in enabling faster training of neural networks, the reasoning behind its effectiveness is not well understood. As a result, various efforts have been made to explain the success of batch normalization. This paper aims to augment these efforts by investigating the effect of batch normalization on the Hessian of the loss function. Namely, we hypothesize that the introduction of batch normalization reduces the condition number of the loss function's Hessian, enabling higher learning rates and faster training. To test this hypothesis, we study a two-layer neural network and investigate the nature of the Hessian with and without the inclusion of batch normalization.

For our analysis, we consider the class of two-layer neural networks used in classification, consisting of a ReLU activation function in the hidden layer followed by a softmax activation in the output layer. The input to the neural network is a vectorized image $x \in \mathbb{R}^n$ (n features) with label $y \in \mathbb{R}^K$ (K classes). The hidden layer with ReLU activation contains m neurons and has output $g(x) \in \mathbb{R}^m$:

$$g(x) = \sigma(W_1 x + b_1), \text{where } \sigma(s) = \max\{s, 0\}$$

The output layer with softmax activation has K neurons and output $f \in \mathbb{R}^K$. Defining $z = W_2 g(x) + b_2$, the output of the network is:

$$f(j) = \text{softmax}(z_j) = \frac{\exp z_j}{\sum_{k=1}^{K} \exp z_k}, \quad j = 1, ..., K$$

1.1. Related Work

Different researchers have suggested possible reasons for the success of batch normalization. The initial motivation for this technique was internal covariate shift, an undesirable phenomenon during training of deep neural networks, wherein the distributions of the inputs to internal layers shift due to parameter updates in previous layers [3]. In batch normalization, the inputs to these internal layers are normalized via rescaling and recentering. While batch normalization was originally claimed to mitigate internal covariate shift, recent research shows that this hypothesized relationship is inaccurate [9].

Another possible explanation for the success of batch normalization is the fact that it "smooths" the optimization landscape, in that it results in greater Lipschitzness of both the loss function and the gradients of the loss function during training [9].

Other researchers attribute the success of batch normalization to its decoupling effect on the parameters of the network. This reasoning is motivated by the success of weight normalization, which is a technique in which the weights of

a neural network are reparametrized in terms of length and direction separately [8]. The result of this decoupling effect is an improvement in the convergence speed of stochastic gradient descent, raising the question whether the success of batch normalization could be attributed to a similar decoupling effect. Indeed, researchers have shown that in the case of gradient descent with batch normalization for the non-convex problem of learning half-spaces, the decoupling effect provably results in an exponential convergence rate [4].

1.2. Contribution and Motivation

In this paper, we aim to reach a better theoretical understanding of the success of batch normalization. In particular, we address the question: *What effect does batch normalization have on the condition number of the loss function's Hessian, and does this effect contribute to its success?*

Our motivation for investigating this condition number stems from the notion of "pre-conditioning" the inputs of a network. When the condition number of the loss function's Hessian is too large, the derivative in one direction increases rapidly while the derivative in another direction increases slowly [2]. The result is poor performance of the gradient descent method, as the learning rate that may work well for one weight differs significantly from the learning rate that works well for a different weight. One remedy is to pre-condition the input data via normalization. Specifically, the input data is rescaled and recentered to be within the unit hypercube [11]. As a result, the conditioning of the Hessian is improved, and faster convergence is achieved [6]. Naturally, this raises the question whether batch normalization of the inputs to the hidden layers of the network improves the Hessian in a similar manner. Our goal is to provide a formal investigation of this question.

2. Theory

This section introduces the relevant theory required to study the effects of batch normalization on network Hessians.

2.1. Batch Normalization

The purpose of batch normalization is to make the output of each layer have unit statistics i.e. $\mathbf{E}[x_i] = 0, \mathbf{var}[x_i] = 1$. The parameters in the lower layers thus do not change the statistics of the input to a given layer. The normalized unit activations are:

$$\hat{x}_i = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

with

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_i^{(j)}, \qquad \sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(x_i^{(j)} - \mu_i)^2$$

and ϵ being very small

These inputs may also be scaled and shifted via trainable parameters γ and β:

$$y_i = \gamma\hat{x_i} + \beta$$

2.2. Hessian Calculation

Letting $\omega \in \mathbb{R}^P$ denote the vector of model parameters, the (i, j)-th element of the Hessian of the loss function L is given by:

$$H_{i,j} = \frac{\partial^2 L}{\partial\omega\partial\omega^\top}$$

In our network of interest, the loss L is the categorical cross-entropy loss. Given N images, each with a ground truth label $y \in \mathbb{R}^K$ and a predicted label $\hat{y} \in \mathbb{R}^K$, the loss function is:

$$\begin{aligned} L &= \sum_{n=1}^{N} L_n(y_n, \hat{y}_n) \\ &= \sum_{n=1}^{N}\left(-\sum_{m=1}^{K} y_{n,m}\log\hat{y}_{n,m}\right) \end{aligned}$$

Our goal is to calculate the Hessian for the network with and without the inclusion of batch normalization. Ideally, we would like to investigate the true Hessian, but obtaining the Hessian is computationally expensive in practice. Consequently, our empirical study employs an approximation of the Hessian, and our analysis is formed through eigenvalue characteristics of the last-layer Hessian.

Forming a block diagonal layer-wise Hessian is a known approximation technique to ameliorate the problem of insufficient memory storage. In fact, it was demonstrated in [1] that the Hessian of two-layer neural network with cross entropy loss converges to a block diagonal matrix. Using this approximation method, the full Hessian of our network is comprised of two block diagonal matrices, the first being size (784×100, 784×100) and the second being size (100×10, 100×10). We strictly analyze the last-layer Hessian for two reasons: (1) memory constraints and (2) it sufficiently represents the effect of batch normalization on the network. In [5], it is noted a general characteristic of the Hessian is that it varies from flat in the first layer to very steep in the last due to the second derivative being very small in the first layers. This provides another theoretical basis for why it is sufficient for only the last-layer Hessian to be analyzed.

3. Empirical Study

The following sections detail the two different network implementations tested on the MNIST Dataset:

3.1. Fully Connected Net

The fully connected network is comprised of two layers: the first layer being a fully connected layer with a ReLU activation function, and the output layer being a fully connected layer using a softmax activation function. The softmax function was chosen in order to normalize the output of the network to a probability distribution. We optimized the network by introducing ℓ_2 regularization. Finally, a range of learning rates and ℓ_2 parameters were iterated through to determine the best performing network.

3.2. Fully Connected Net with Batch Normalization

For the network of interest, we utilized the same structure as in Section 3.1 but introduced batch normalization after the ReLU layer. Here, the same learning rate and ℓ_2 parameters were used.

Note that although Ioffe and Szegedy initially recommended that the batch normalization layer should be added before the nonlinearity [3], empirical results have shown that adding the batch normalization after the hidden layer activation performs the same or better [10]. As a result, we place the batch normalization layer after the ReLU layer in our network.

4. Results

Each network was optimized for performing predictions on the MNIST training set. Adam was used as the optimizer for the benefits of having adaptive learning rates and momentum. Meanwhile for the loss function, Cross Entropy Loss was used due to this being well-suited for classification tasks. The following three evaluation points were performed: (1) Prediction Accuracy, (2) Convergence Rate, and (3) Hessian Analysis.

4.1. Prediction Accuracy

The Batch Normalized network showed an improvement in accuracy for both training and testing data. Because the testing accuracy is similar to training, we see that the networks generalize well.

Network	Training	Validation	Testing
FC	74.8%	77.2%	72.8%
FC + BN	79.5%	81.3%	76.7%

Table 1. Network Prediction Accuracy

4.2. Convergence Evaluation

As anticipated, the fully connected network with batch normalization had a faster convergence rate in comparison to the same network without any batch normalization. These results are shown in Figures 1 and 2. The time axis refers to the world clock, with the number of epochs being 30 for both networks. As shown, the time per epoch is greater for the batch normalized network due to computational time for completing normalization, yet still leads the network to faster convergence overall.

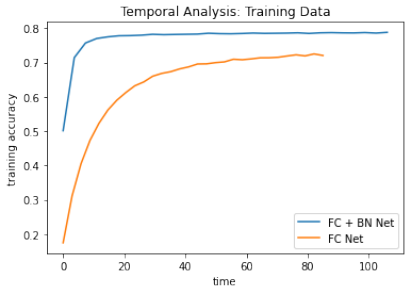

Figure 1. Temporal Evaluation on Training Data

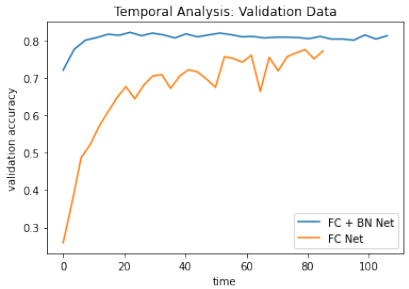

Figure 2. Temporal Evaluation on Validation Data

4.3. Hessian Analysis

The Hessian of the loss function with respect to the weights of the last layer, W_2 was calculated exactly utilizing Tensorflow. The last-layer Hessian matrix for both the batch normalized and baseline case had several eigenvalues on the order of $10^{-0.9}$. In our hypothesis, we expected to directly compare the condition number, $\kappa = \lambda_{\max}/\lambda_{\min}$, between the two networks. However, because the minimum eigenvalue is close to zero, this causes the condition number κ to be very large and unsuited for a comparison metric.

In our experimental results, we noticed there to be a significant difference in the maximum eigenvalues of the batch-normalized network versus the un-normalized network. A pseudo-condition number, $\kappa' = \lambda_1/\lambda_{10}$ where λ_1 is the largest eigenvalue and λ_{10} is the 10th largest eigenvalue, was instead used as the metric to compare the networks shown in Figure 3.

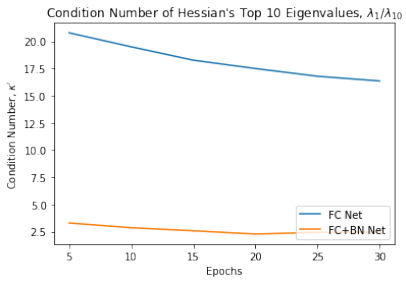

Figure 3. Hessian Eigenvalue Evaluation on Training Data

From Figure 3 we see the pseudo-condition number is much larger for the network without batch normalization, meaning the the top ten eigenvalues are both larger and have a wider distribution in comparison to with normalization. The batch-normalized network returning consistently smaller maximum eigenvalues for the loss function's Hessian could be a possible explanation for the fast convergence.

5. Discussion

In response to the unexpected singularities in our Hessian matrices, we conducted further literature review. Other researchers have also found that the Hessians of certain neural networks have eigenvalues concentrated near zero [12]. In fact, Sagun et al. [7] studied a similar two-layer network with cross-entropy loss, and found that the Hessian is singular and that the degree of singularity increases with the number of units in the hidden layer. This would be a suitable explanation for why many of the eigenvalues in our network are close to zero, leading to a large condition number. Perhaps as a consequence of these singularities, various Hessian-based analyses on neural networks, such as in [13], present results for a so-called Hessian spectrum instead of a condition number. The Hessian spectrum can be described as how the top k eigenvalues evolve through training epochs. Moreover, it has been observed that networks used in classification often have c large eigenvalues, where c is the number of classes [12], further supporting our construction of a pseudo-condition number using the ten largest eigenvalues.

6. Conclusion

This work contributes to the long-standing investigation of why batch normalization in neural networks leads to a faster convergence rate. Specifically, we examine the Hessian of the loss function with respect to the weights of networks with and without batch normalization in order to determine a correlation to faster convergence. We found that the condition number may not be an appropriate metric for the effects of batch normalization on a network with many parameters, as the Hessian is often singular. Nonetheless, the experimental results show that the maximum eigenvalues of the Hessian between the batch normalized and control network have a significant gap, demonstrating an improved pseudo-condition number. We propose that the success of batch normalization can be attributed to this improved pseudo-condition number, allowing for faster convergence rates. Further theoretical work is required to prove this relationship and to generalize these empirical results to other networks.

References

[1] R. Collobert. Large scale machine learning. Technical report, Université de Paris VI, 2004.

[2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[4] J. Kohler, H. Daneshmand, A. Lucchi, T. Hofmann, M. Zhou, and K. Neymeyr. Exponential convergence rates for Batch Normalization: The power of length-direction decoupling in non-convex optimization. *AISTATS 2019 - 22nd International Conference on Artificial Intelligence and Statistics*, 2020.

[5] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[6] G. Montavon, G. G. B. Orr, K.-R. Müller, Y. LeCun, L. Bottou, G. G. B. Orr, K. Muller, G. Montavon, G. G. B. Orr, and K.-R. Müller. *Neural Networks: Tricks of the Trade*. Number MAY 2000. 2012.

[7] L. Sagun, L. Bottou, and Y. LeCun. Eigenvalues of the Hessian in Deep Learning: Singularity and Beyond. pages 1–8, 2016.

[8] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[9] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 2018-December(NeurIPS):2483–2493, 2018.

[10] F. Schilling. The Effect of Batch Normalization on Deep Convolutional Neural Networks. page 113, 2016.

[11] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3 PART 3):1464–1468, 1997.

[12] Y. Wu, X. Zhu, C. Wu, A. Wang, and R. Ge. Dissecting Hessian: Understanding Common Structure of Hessian in Neural Networks. 2020.

[13] Z. Yao, A. Gholami, Q. Lei, K. Keutzer, and M. W. Mahoney. Hessian-based analysis of large batch training and robustness to adversaries.

Appendices

A. Accuracy and Loss Convergence Evaluation

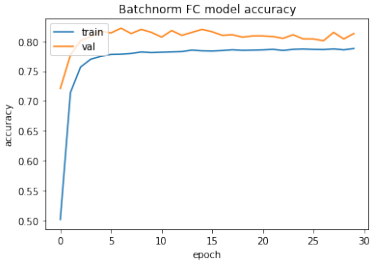

Figure 4. Two Layer Network BatchNorm Accuracy

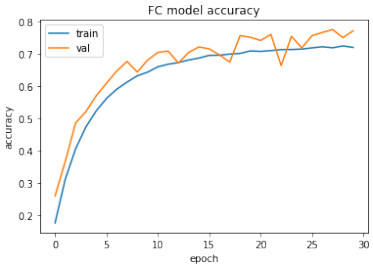

Figure 5. Two Layer Network Accuracy

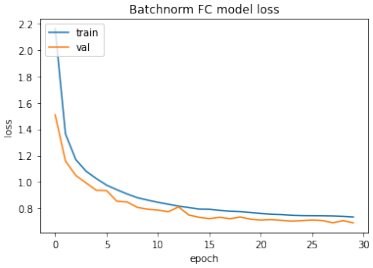

Figure 6. Two Layer Network BatchNorm Loss

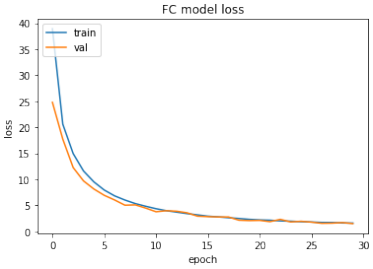

Figure 7. Two Layer Network Loss

B. Hessian Evaluation

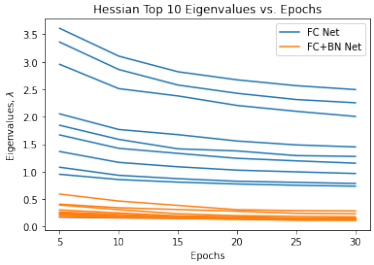

Figure 8. Largest 10 Eigenvalues of Hessian