Understanding Batch Normalization

Depen Morwani

July 2020

1 Background

Batch Normalization is one of the most important innovations in Deep Learning architectures, that leads to easy trainability of deep networks for a variety of hyperparameters. Although the original paper suggested that the primary reason why batch normalization works is reducing internal covariate shift, the exact theoretical reasons for its effectiveness are not well understood.

One of the first papers that tried to address this issue was by Santurkar et al. (2018). It showed that even when the distribution of inputs at internal layers is explicitly modified by adding some noise, the network is still able to learn. They also showed that batchnorm leads to smoothing of the optimization landscape.

Bjorck et al. (2018) tried to empirically establish that the primary reason why Batchnorm leads to higher testing accuracies is by enabling higher learning rates. It showed, for 110-layer Resnet, that BN and non-BN network trained at same LR, lead to similar testing accuracies, but BN networks can be stably trained for a much higher lr, which is the primary reason for better accuracy.

Kohler et al. (2019) was one of the first papers that explicitly tried to analyze the optimization trajectory of batchnorm. It showed, under certain assumptions, that BN leads to linear convergence for an ordinary least squares(OLS) as well as for Generalized Linear Models(GLM), with an adaptive step size schedule.

However, the above paper could not shed light on why BN enables higher lr. Cai et al. (2019) addressed this issue exactly in case of OLS, by showing that with BN reparameterization, gradient descent converges for any lr upto 1. This is very different from the general case, where the maximum lr allowed is $\frac{2}{\lambda_{max}}$, where λ_{max} denotes the maximum eigenvalue of the Hessian.

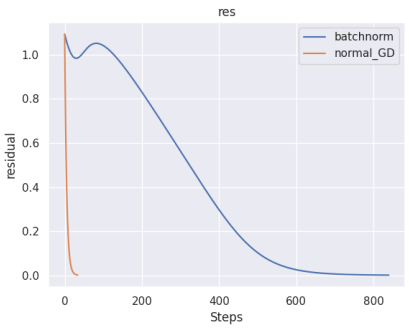
Arora et al. (2019) established that the convergence rate for GD/SGD on a batch-normalized neural network is unaffected by the learning rate used for scale-independent parameters. Hoffer et al. (2018) showed that other normalizations such as L_1 or L_∞ can also replace batch normalization and can be used for low-precision implementations of networks.

In this note, we would primarily focus on empirical experiments related to how Batch Normalization affects the parameter path that the network takes, along with its interaction with noise in the dataset.

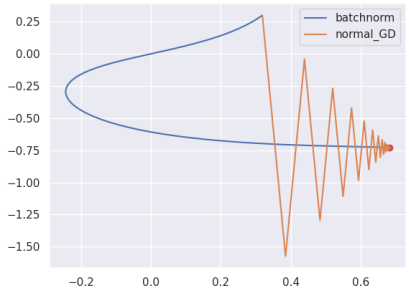
2 OLS

Consider the case of an ordinary least squares problem, i.e, min. $E[(y - x^T w)^2]$, where $E(xy) = g$ and $E(xx^\top) = H$. Denote $u = H^{-1}g$ In case of standard gradient descent, it's already known that the maximum allowed step size is $\frac{2}{\lambda_{max}}$, where λ_{max} represents the maximum eigenvalue of H.

If we use BN, then the output would be $\gamma * \frac{x^\top w - E(x^\top w)}{\sqrt{Var(x^\top w)}}$, which under the assumption $E(x) = 0$, transforms to $\gamma * \frac{x^\top w}{\sqrt{w^\top H w}}$. Thus, with BN transformation, the OLS problem becomes $E[(y - \gamma * \frac{x^\top w}{\sqrt{w^\top H w}})^2]$, where γ and w are the parameters that need to be optimised. Thus, BN can be thought as a reparameterization of the problem in approximately polar coordinates, with the weight given by $\frac{\gamma * w}{\sqrt{w^\top H w}}$.

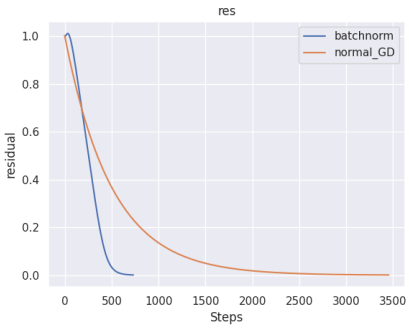


(a) Residual

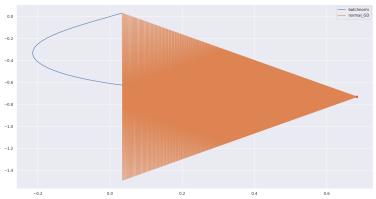


(b) Weight trajectory

(c) $\kappa = 10$



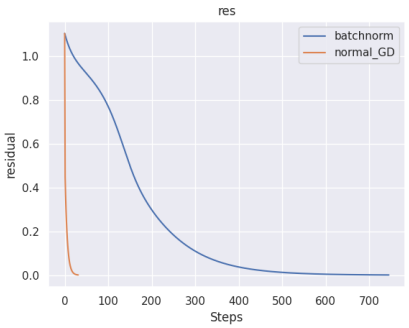
(d) Residual



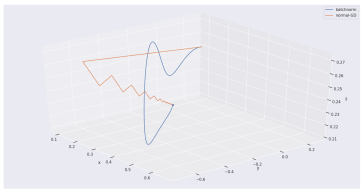
(e) Weight trajectory

(f) $\kappa = 1000$

d=2

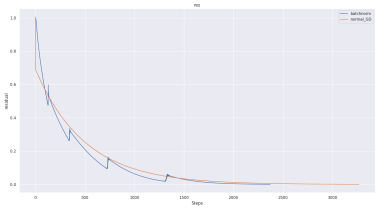


(a) Residual

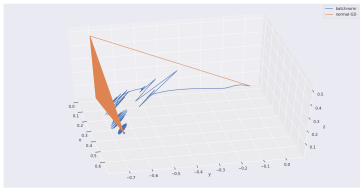


(b) Weight trajectory

(c) $\kappa = 10$



(d) Residual



(e) Weight trajectory

(f) $\kappa = 1000$

d=3

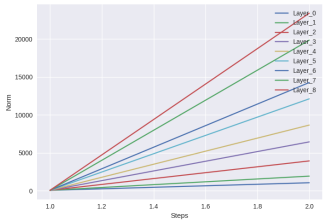
Experiments were conducted for comparing trajectories and convergence speed for OLS with and without BN, in dimension 2 and 3 where trajectories could be visualised(κ represents the condition number of H).

One observation from the experiments was that the rate of convergence of BN is almost unaffected by condition number in d=2. This is in line with the arguments of Cai et al. (2019) as they show that the asymptotic convergence speed with batchnorm depends on the pseudo-condition number of $H^* = H - \frac{gg^\top}{u^\top Hu}$, which is independent of κ of H for d=2 . However, in higher dimensions, we see for small condition numbers, non-BN networks converge faster, while for higher condition numbers, the convergence rate of BN is slightly better than non-BN networks.

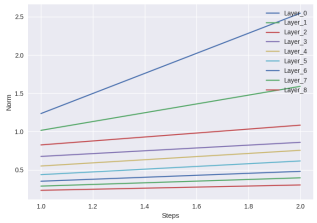
3 Why does BN enable higher lr ?

In this section, we would empirically establish why BN enables higher lr. First of all, we define the concept of an equivalent batch-norm network. Consider, a BN forward pass, which is given by $\gamma * \frac{w^\top x + b - \mu}{\sigma} + \beta$. We can define $w_{eq} = \frac{\gamma}{\sigma} * w$ and $b_{eq} = \frac{\gamma}{\sigma} * (b - \mu) + \beta$, so that the batchnorm operation could be written as

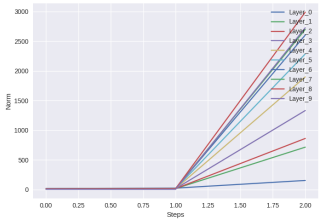
$w_{eq}^{\top} x + b_{eq}$.



(a) Activations norm

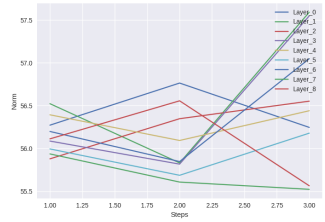


(b) Activations gradient norm

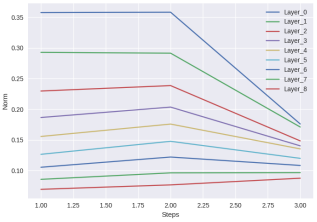


(c) Weight norm

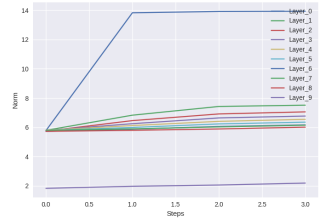
Training of Non-BN network



(a) Activations norm



(b) Activations gradient norm



(c) Weight norm

Training of BN network

In the experiments above, we compare a BN and non-BN neural network with 10 layers, trained on 10000-subsampled MNIST dataset using batch gradient descent and lr=1, with the non-BN network initialized at the equivalent point of the batch norm network. lr=1 has been chosen as BN doesn't diverge at this lr, but the neural net without BN diverges.

3.1 Explanation

As can be seen from the above experiment, at higher lr, the activations and weights of non-BN network diverge, but thats not the case for BN network. The explanation lies in the forward as well as backward normalization of BN.
Consider a layer without BN, that for an input batch Y_{l-1} produces an output Y_l, i.e, $Y_l = WY_{l-1} + B$. Let's say the gradient of Loss with respect to Y_l is denoted by $\frac{\partial L}{\partial Y_l}$. Then, by chain rule,

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y_l} Y_{l-1}^T$$

$$\frac{\partial L}{\partial Y_{l-1}} = \frac{\partial L}{\partial Y_l} W$$

This explains the three-fold reason behind the activation and gradient explosion for non-BN network :

1. As the norm of W grows, the norm of Y grows.
2. As the norm of W grows, the norm of $\frac{\partial L}{\partial Y_{l-1}}$ grows.
3. As the norm of Y_{l-1} grows, the norm of W grows.

Thus, the weights norm explosion, backpropagated gradient explosion and activation explosion, are all positively correlated with each other, which leads to sudden divergence as depth increases(due to exponential growth with depth).

For a BN network, the forward propagation is given by $Y_l = \gamma * \frac{w_i^T Y_{l-1} + b_i - \mu}{\sigma} + \beta$(we only focus on one neuron as notation becomes hairy for multiple neurons, but the results are similar, also notice Y_l is a vector while Y_{l-1} is a matrix). Let's denote $Z_l = \frac{w_i^T Y_{l-1} + b_i - \mu}{\sigma}$. In this case, the backpropagated gradients are given by

$$\frac{\partial Y_l}{\partial Z_l} = \frac{1}{\sigma}(I - \frac{1}{n}11^T - Z_l Z_l^T)$$

Then, using chain rule, we get

$$\frac{\partial L}{\partial w_i} = \frac{\gamma}{\sigma}(I - \frac{1}{n}11^T - Z_l Z_l^T) Y_{l-1}^T \frac{\partial L}{\partial Y_l}$$

$$\frac{\partial L}{\partial Y_{l-1}} = \frac{\gamma}{\sigma}(I - \frac{1}{n}11^T - Z_l Z_l^T) \frac{\partial L}{\partial Y_l} w_i^T$$

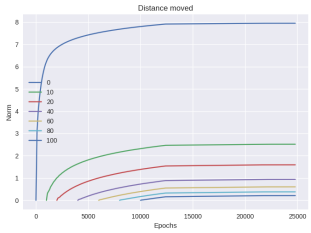
BN prevents backpropagated gradients from exploding as follows :

1. Y does not grow as W grows, due to forward normalization.
2. $\frac{\partial L}{\partial Y}$ does not grow as W grows as it has σ in the denominator, which approximately cancels out the effect of w_i term in numerator
3. γ and β do not blow up as $\frac{\partial L}{\partial Y}$ does not blow up.

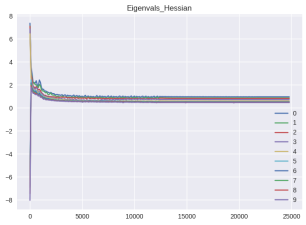
These are the primary reasons why backpropagated gradients and activations do not blow up in case of batchnorm.

4 Other normalizations and smoothness

The properties outlined above responsible for enabling higher lr, are satisfied by any normalization, whether it be L_1, L_3 or L_∞, centered or not. Thus, every normalization enbales higher lr, which has been verified in previous papers as well(Santurkar et al., 2018; Hoffer et al., 2018). But, there has been no comparison regarding which normalization may be best from generalization perspective. From recent studies on generalization bounds(Li et al., 2020), it seems as if the Hessian eigenvalues at optimization, i.e, the smoothness along with distance from initialisation can be a potential indicator of generalization. Following the same, we evaluated top Hessian eigenvalues and distance from initialisation for multiple normalization techniques on 10000-subsampled MNIST dataset using Lanczos algorithm(Golmant et al., 2018).

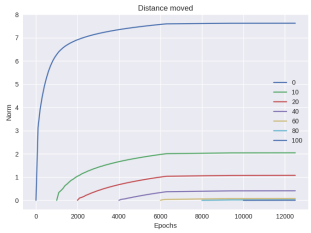


(a) Distance moved

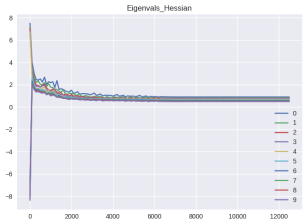


(b) Top 10 Hessian eigenvalues

centered L_2 normalization : Best val acc - 96.37%

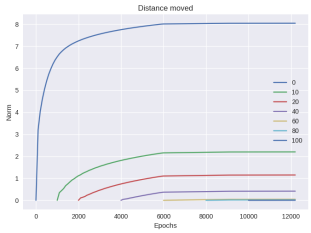


(a) Distance moved

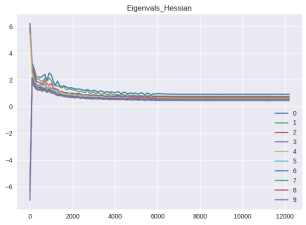


(b) Top 10 Hessian eigenvalues

uncentered L_2 normalization : Best val acc - 96.06%

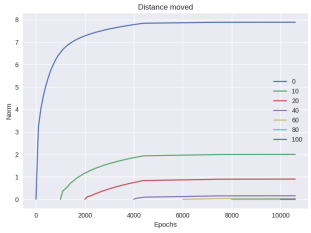


(a) Distance moved

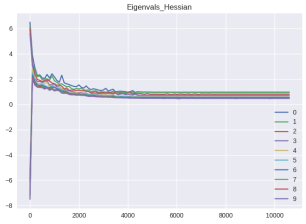


(b) Top 10 Hessian eigenvalues

centered L_3 normalization : Best val acc - 96.24%



(a) Distance moved



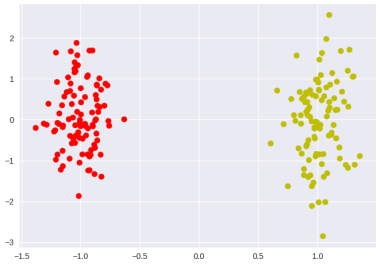
(b) Top 10 Hessian eigenvalues

uncentered L_3 normalization : Best val acc - 96.02%

We can see that although there is a slight difference in the best validation accuracy reached, but the final eigenvalues and distance from initialization is comparable. Also, the difference in best validation accuracy is not significant.

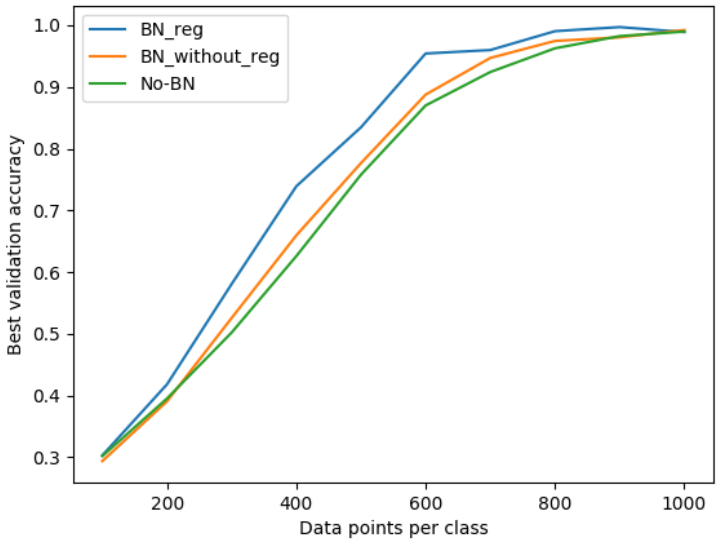
5 Robustness to noise

Most of the modern world CV datasets are high-dimensional and thus contain enormous number of features, some of which may be useful just on the training dataset. To replicate these properties, a custom dataset is created in which the inputs are d-dimensional, out of which $d1$ dimensions are useful, where data is linearly separable for the classes and the remaining dimensions are random Gaussian noise. An example 2-dimensional dataset is shown, where the classes are separable along the first axis and second axis values are just noise.



2-d dataset

We test Batchnorm and non-batchnorm network with identical architecture(1 hidden layer net with 100 neurons) on an instance of this dataset with $d = 1000$, $d1 = 100$ and 5 classes. The number of training data points for each class is varied and the corresponding best validation accuracy is plotted.(BN_reg means examples are shuffled across each epoch so that BN has a regularization effect, BN_without_reg means examples are not shuffled)



Best validation accuracy

We can see batchnorm, even without regularization, beats the non-batchnorm network in terms of best validation accuracy, when there is less training data.

6 Future work

1. Understand theoretically the smoothness effect of various normalization techniques.
2. Theoretically understand BN robustness to noise

Références

Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rkxQ-nA9FX`.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7694–7705. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7996-understanding-batch-normalization.pdf`.

Yongqiang Cai, Qianxiao Li, and Zuowei Shen. A quantitative analysis of the effect of batch normalization on gradient descent. volume 97 of *Proceedings of Machine Learning Research*, pp. 882–890, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/cai19a.html.

Noah Golmant, Zhewei Yao, Amir Gholami, Michael Mahoney, and Joseph Gonzalez. pytorch-hessian-eigentings : efficient pytorch hessian eigendecomposition, October 2018. URL https://github.com/noahgolmant/pytorch-hessian-eigenthings.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters : efficient and accurate normalization schemes in deep networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2160–2170. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7485-norm-matters-efficient-and-accurate-normalization-schemes-in-deep-networks.pdf.

Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Thomas Hofmann, Ming Zhou, and Klaus Neymeyr. Exponential convergence rates for batch normalization : The power of length-direction decoupling in non-convex optimization. volume 89 of *Proceedings of Machine Learning Research*, pp. 806–815. PMLR, 16–18 Apr 2019. URL http://proceedings.mlr.press/v89/kohler19a.html.

Xinyan Li, Qilong Gu, Yingxue Zhou, Tiancong Chen, and Arindam Banerjee. Hessian based analysis of sgd for deep nets : Dynamics and generalization. In Carlotta Demeniconi and Nitesh Chawla (eds.), *Proceedings of the 2020 SIAM International Conference on Data Mining, SDM 2020*, Proceedings of the 2020 SIAM International Conference on Data Mining, SDM 2020, pp. 190–198. Society for Industrial and Applied Mathematics Publications, 2020. doi : 10.1137/1.9781611976236.22. 2020 SIAM International Conference on Data Mining, SDM 2020 ; Conference date : 07-05-2020 Through 09-05-2020.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization ? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2483–2493. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf.