STAD80 Final Project: Research on ideologies of Batch Normalization

Danhua Cai, Siyi Wei
April 20, 2020

1 Introduction

Batch Normalization(BN) is a convenient and resource friendly regularization method in training deep neural network. Empirically it could save more time in converging to the same loss compare to not using batch normalization during training. However, in the original paper[1] it barely discusses the ideology behind batch normalization, the author raised one hypothesis: Batch normalization could reduce internal co-variate shift(ICS), which could lead to a fluctuation of the hidden layers' distribution during training. We reviewed five relevant papers and use the collected evidence to substantiate a different point of view. **Internal co-variate shift is not directly related to slowing down the process of training. And Batch Normalization also could not guarantee to reduce Internal co-variate shift. Instead, Batch normalization could boost this process through a more fundamental reason.**

2 Backgrounds of Batch Normalization and Internal Co-variate Shift

2.1 Internal Co-variate Shift

Internal Co-variate Shift was commonly agreed to be one obstruct in accelerating the training process. It was defined as the change of the distribution in hidden layers between training steps. Since each network layer could be think as an independent optimization problem, it is essential to keep the input for this optimization problem to be stable. In other word, a stable input distribution will led to a robust optimized model. On the contrast, if the model has significant internal co-variate shift during training. It would be common to requires more time to have a stable convergence. One little example is to think of a classification problem with the following decision bound. Where the red are the train examples and the blue are the test examples. Even though the distributions between train set and test set are different. It would be reasonable if they were classified with the same label since they are in the circle decision boundary. Here we could use normalization to shift dataset so they have the same distribution, which lead to accelerating the training process. One question may be raised as what if the sector below has different labels? In other word, what if the input distribution should be shift? That's the reason why we need learnable parameters γ, β when we introduce Batch Normalization.

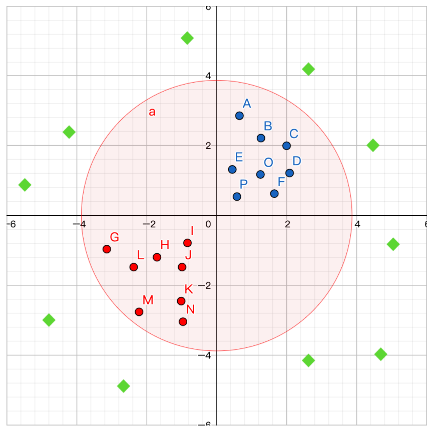

Figure 1: DataSet Before Normalization

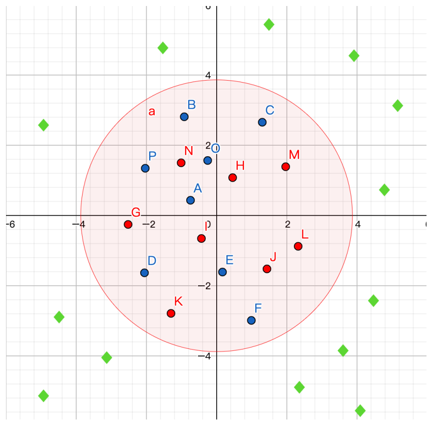

Figure 2: DataSet After Normalization

2.2 Batch Normalization

The algorithm for Batch Normalization is simple. In deep learning we split the training sets into different small batches where we could train them batch by batch. Such technic is called batch gradient descent. In gradient descent we need to normalize the data before training. Such normalization method has a lot advantages. For example, it could improve the optimization process by having relatively small variance and mean for batches. It would be sensible to have such thought, in deep learning each hidden layer can be treated as an optimization problem. Will the network be benefit if we add normalization method before each hidden layer? The empirical answer is yes: If we could apply such normalization method correctly.

Input: Values of x (Neuron Input) over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$; Learnable parameters: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i, \qquad \text{(mini-batch mean)}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2, \qquad \text{(mini-batch variance)}$$

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \qquad \text{(normalize)}$$

$$y_i \leftarrow \gamma \widehat{x_i} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i), \qquad \text{(scale and shift)}$$

Algorithm: Batch Normalizing Transform, applied to activation x over a mini-batch.

Back to our example in Figure 1 and Figure 2. For each activation $x^{(k)}$ has a pair of parameters $\gamma^{(k)}$ and $\beta^{(k)}$ which can learned along the original model parameters. This pair of learnable parameters gave the distribution the ability to shift. In fact, by setting $\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$ and $\beta^{(k)} = \mathbf{E}[x^{(k)}]$, we could recover the original distribution.

3 How does Batch Normalization help Optimization?

3.1 Disagreement on original point of view

In the orginal article Sergey Ioffe and Christian Szegedy assume batch normalization could reduce ICS thus improve the training process. However, in a recently published paper at 2019[2]. Shibani.S, Dimitris.T, Andrew.I and Aleksander.M raised a different point of view for batch normalization. They think batch normalization helps the optimization problem to be robust by smoothing the loss function rather than reducing internal co-variate shift. They substantiate their point of view by proving the relation between ICS and the improvement of training is tenuous. Moreover, they gave the counter example that under specific situation, batch normalization could increase internal co-variate shift.

3.2 Relation between ICS and improvement of training

The authors prove there is no relation, or at least this relation is tenuous, between the reduction of internal co-variate shift and the improvement in training deep neural network. Their method is adding random non-zero mean noise between hidden layers. It could increase the internal co-variate shift manually. Surprisingly the steps for loss function need to converge was not significantly reduced, there is still significant improvement than the vanilla network. We re-implement such experiment on different dataset and got the similar result, this result could prove there is at least tenuous relation between internal co-variate shift and improvement in optimization.

3.3 Relation between BN and ICS

Since internal covariate shift was not clearly defined in the original article. The authors then define the way to measure ICS using the change of the gradient for the loss function:

Definition 3.1. Let $\mathcal{L}$ be the loss, $W_1^{(t)}...W_k^{(t)}$ be the parameters of each of the k layers and $(x^{(k)}, y^{(k)})$ be the batch of input-label pairs used to train the network at time t. We defind internal covariate shift(ICS) of activation i at time t to be the difference $||G_{t,i} - G'_{t,i}||_2$ where

$$G_{t,i} = \nabla_{W_i^{(t)}} \mathcal{L}(W_1^{(t)}, ..., W_k^{(t)}; x^{(k)}, y^{(k)})$$
$$G'_{t,i} = \nabla_{W_i^{(t)}} \mathcal{L}(W_1^{(t+1)}, ..., W_{i-1}^{(t+1)}, W_i^{(t)}, W_{i+1}^{(t)}, ..., W_k^{(t)}; x^{(k)}, y^{(k)})$$

$G_{t,i}$ corresponds to the gradient of the layer parameters that would be applied during a simultaneous update of all layers (as is typical). On the other hand,$G'_{t,i}$ is the same gradient after all the previous layers have been updated with their new values. The difference between G and G thus reflects the change in the optimization landscape of W_i caused by the changes to its input.

The authors indicate that the The difference between G and G' captures precisely the effect of cross-layer dependencies that could be problematic for training. They claim that there is at least tenuous relation between batch normalization and the internal co-variate shift they defined. What's more, They have found there for particular situation batch normalization could even lead to more internal co-variate shift. However, the experiment is arguable and different opinions were raised by You Huang, Yuanlong Yu in 2020.[4]. They think the article defined an unsuitable ICS measure using the difference between gradient because the gradients are too sensitive.

You Huang and Yuanlong Yu raised a different ICS measurement using the Earth Mover distance and derives the upper and lower bounds for ICS. And they prove that batch normalization could provide bounds for ICS. However, a reduction of ICS is not yet been proven thus the more fundamental ideology of batch normalization remains unclear.

3.4 Relation between BN and the smoothness of loss function

The authors[2] who disagree the original point of view for BN then raise a different hypothesis on how BN improve the optimization problem. Batch Normalization could smooth the loss function significantly. A smoother loss function is superior compare to non-smooth loss functions in gradient based optimization problem, which was also been visualized by Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer and Tom Goldstein in their published article[4].

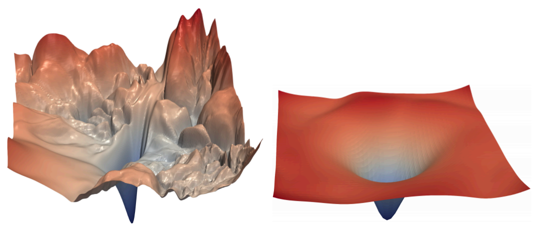

Figure 3: Non Smooth VS Smooth loss landscape

Moreover, the authors proved Batch Normalization could affect the Lipschitzness of the loss function. By equipment of the definition for β-Lipschitzness, and β-Smootheness.

Definition 3.2. (β-Lipschitz) Let $C \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is β-Lipschitz over C if for every $w_1, w_2 \in C$ we have that $\|f(w_1) - f(w_2)\| \leq \beta||w_1 - w_2||$

Definition 3.3. (β-Smooth) A differentiable function $f : \mathbb{R}^d \to \mathbb{R}^k$ is β-Smooth if its gradient is β-Lipschitz; namely, for all v, w we have $\|\bigtriangledown f(v) - \bigtriangledown f(w)\| \leq \beta||v - w||$

The authors[2] begin with considering the optimization landscape with respect to the activation y_i. By focusing on the gradient magnitude $|| \bigtriangledown_{y_i} L||$, which captures the Lipschitzness of the Loss. The Lipschitz constant of the loss plays a crucial rule in optimization as it described the value which the loss change when taking a step. They then showed using batch normalized landscape exhibits a better Lipschitz constant. More than that, they showed the Lipschitz constant is significantly reduced whenever the activations $\widehat{y_j}$ correlate with the gradient $\bigtriangledown_{\widehat{y_i}} \widehat{L}$ or the mean of the gradient deviates from 0. This effect hold even the scaling of batch normalization is identical with the original distribution. The authors therefore proved batch normalization could produce steps are more predictive than vanilla network with respect to give a smaller Lipschitz constant by proving the following theorems:

Theorem 3.4. (The effect of Batch Normalization on Lipschitzness of loss) For a BatchNorm network with loss $\widehat{L}$ and an identical non-BN network with identical loss L,

$$|| \bigtriangledown_{\widehat{y_i}} \widehat{L}|| \leq \frac{\gamma^2}{\sigma_j^2}(|| \bigtriangledown_{y_i} L||^2 - \frac{1}{m}\langle 1, \bigtriangledown_{y_i} L\rangle^2 - \frac{1}{m}\langle \bigtriangledown_{y_i} L, \widehat{y_i}\rangle^2).$$

Theorem 3.5. (The effect of Batch Normalization on Smoothness of loss) Let $\widehat{g_j} = \bigtriangledown_{y_j} L$ and $\mathbf{H}_{jj} = \frac{\partial L}{\partial^2 y_j}$ be the gradient and Hessian of the loss with respect to the layer outputs respectively, Then

$$(\bigtriangledown_{\widehat{y_i}} \widehat{L})^T \frac{\partial \widehat{L}}{\partial^2 y_j}(\bigtriangledown_{\widehat{y_i}} \widehat{L}) \leq \frac{\gamma^2}{\sigma_j^2}(\frac{\partial \widehat{L}}{\partial y_j})^T \mathbf{H}_{jj}(\frac{\partial \widehat{L}}{\partial y_j}) - \frac{\gamma}{m\sigma^2}\langle \widehat{g_j}, \widehat{y_j}\rangle || \frac{\partial \widehat{L}}{\partial \widehat{y_j}}||^2$$

3.5 Use LP norm to reach the same effect with Batch Normalization

By proving batch normalization could smooth the loss landscape. It would be reasonable to ask: Could other smoothing methods reaching the same effect compare to batch normalization. Thus the authors compared the result by using batch normalization and LP normalization. Then conclude they reached the similar performance on improving the training process, where this finding could substantiate their hypothesis. Batch normalization improve training process by smoothing the loss function.

4 Similar technics that could help optimization

4.1 Layer Normalization

Beside all the advantages of batch normalization. It also have some common disadvantages. For example, it requires the batch size to be relatively large since normalize small batch might decrease the accuracy of the distribution. Batch normalization is also not suitable to Recurrent Neural Network. It would remove the moving trend of the distribution, also know as the sequence trend. Alternatively there is another method could replace batch normalization, such technic is called Layer Normalization which was introduced by Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton in 2016.[3]

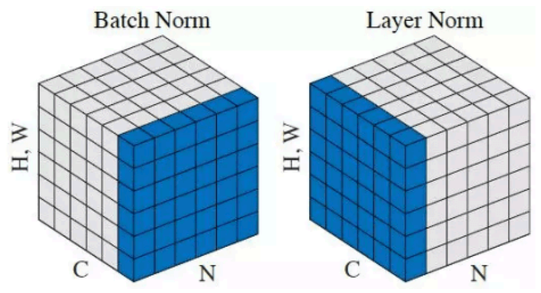

Figure 4: Difference between Batch Norm and Layer Norm

The ideology behind Layer Normalization is simple. We normalize the batch along layers. So the relation between each sample will still be kept. This is essential to RNN, while we normalize along the layers, the sequence trend would be kept. Layer normalization also helps when the batch size is too small. Since layer normalization relies on amount of neurons connected to normalization layer instead of the batch size. In online learning, where the batch size is often small. Layer optimization will be more practical.

5 Conclusion

After we organized relevant papers of batch normalization. We found the ideology of batch normalization remains unclear and arguable. Among all the possible hypothesises, we agree more on batch normalization could help improving training by smoothing the loss landscape. This agreement was substantiate by our understanding on the ideology of regularization. However, the current proof[2] is yet to be perfect.
Moreover, we organized several alternative method using for normalization. Each of them has its own pros and cons. The research on the ideology of how normalization regularize neural network is still valuable.

References

[1] Sergey Ioffe, Christian Szegedy, *arXiv: 1502.03167.* Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015.

[2] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry, *arXiv: 1805.11604.* How Does Batch Normalization Help Optimization? 2018.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, *arXiv: 1607.06450.* Layer Normalization, 2016.

[4] You Huang, Yuanlong Yu, *arXiv: 2001.02814.* An Internal Covariate Shift Bounding Algorithm for Deep Neural Networks by Unitizing Layers' Outputs, 2020.

[5] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein, *arXiv: 1712.09913.* Visualizing the Loss Landscape of Neural Nets. 2018.