ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 8, August 2017

Batch Normalization and Its Optimization Techniques: Review

[1] Niveditha Kumaran, [2], Ashlesha Vaidya
[1][2] Department of Computer Science Engineering
[1][2]SRM University, Chennai

***Abstract*— Batch normalization is a boon to the training of a deep neural network. It acts as a panacea to the problem of internal covariate shift and facilitates the usage of higher learning rates. It also accounts for the inclusion of saturating non-linear functions, while excluding the need of drop outs for regularisation. However, mini-batch normalization is not self-sufficient and comes with a few limitations such as inability to deal with non-i.i.d inputs and decreased efficiency with a batch size of one. In this paper, we explore normalization, the need for its optimization, and evaluate the optimization technique provided by researchers.**

***Index Terms*— i.i.d input, non-linear, normalization, internal covariate shift.**

I. INTRODUCTION

Deep neural networks are an efficacious model, for acute classification, clustering, regression, segmentation and, various other functionalities. With the increase in their depth, a number of complex computations can be accounted for.Various algorithms exist to train such deep networks, and one such efficient implementation is the backpropagation, using Stochastic Gradient descent [1]. It works to plummet the loss, by updating parameters consistently. But it also necessitates appropriate initialization of values and optimum learning rates. Moreover, in a network so deep, every layer's input is influenced by the result of the previous activation [12].

Hence, even a nuance in the initial stage magnifies as it propagates towards the deeper ones. These changes alter the domain of the activation functions, which implicitly lead to the extraction of unanticipated features, that leave the network perplexed. This is known as internal covariate shift, which is an undesired property [2]. Normalization of a network helps abate such problems.

Batch normalization produces unit variance and zero mean, which control the spread of individual inputs to all the layers in a net, thus stabilizing the model to be trained with ease [4], [5]. Also, with the ability to handle local minima in the process of loss minimization, the network can now implement saturating non-linearities with ease, and also account for a higher learning rate thus accelerating training as a whole.

Although Batch normalization seems to be a cure all for deep networks, it comes with the inability to account for a micro-batch size of one, and non- i.i.d inputs [6]. Moreover, the elements of a batch, after normalization are dependent on its core batch, and hence turn out to be biased, or overfit the specific, hence calling for an optimization [9]. In this paper, we explore basics of batch normalization algorithm, its limitations and analyze the efficiency of its optimization technique known as batch re-norm, proposed by researchers for enhanced results.

II. BATCH NORMALIZATION

2.1 Algorithm

Batch normalization is based on two basic concepts: normalization of every dimension autonomously (i.e. every feature), and distribution of inputs across mini-batches, which are later interfaced as a single unit [5].
For every feature, across every batch the following steps are computed:
Where x_i depicts input of each x from the mini-batch of size m, μ_B is the mean over the specific batch, σ_B^2 is the batch variance and y_i is the output corresponding to each input in batch .

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

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

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

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 8, August 2017

$$y_i = \gamma \hat{x}_i + \beta$$

γ and β are known as scale and shift parameters which are later learned through back propagating the network for each batch.

Once this process is complete, for each feature, including every batch, parameters γ, β, and the weights are trained through back propagation.

Now, the individual batches are interfaced into a single unit through the following steps:

For every feature:

The mean of all the μ_B s accounting to every batch is:

$$\mu = mean(\mu_B)$$

For every activation, the mean of variances across every batch B, with size m:

$$\sigma = \frac{m}{m-1} mean(\sigma_B^2),$$

And finally, by replacing

$$y = \frac{\gamma(x-\mu)}{\sqrt{\sigma + \in}} + \beta$$

batch normalization completes.

2.2 Advantages of the process

It prevents the gradients from entering the region of local minima, by rectifying the variations and extremities [11]. It normalizes the parameters and also ensures that the gradient remains intact, without vanishing or diminishing. This helps secure the parameters from exploding, hence enabling the use of higher learning rates, as compared to the case otherwise [5].

The training process takes into account a number of batches and finds out the normalized features to be checked for at each transaction. And since every batch consists of a number of samples, the model is never overfit for a particular example, or biased to a particular type of input, thus removing the need for a process called regularization as a whole. Hence, without the use of drop outs [3], batch normalization attains accuracy tantamount to that.

III. PROBLEMS WITH NORMALIZATION

Initial algorithm derives the activations of a particular batch based on the other examples in the set. In other words, the rest of the members of a batch are solely responsible for influencing the various parameters in it. This characteristic of being slightly biased to a particular batch poses a problem when interfacing, along with other batches.

In the algorithm, the batches are trained using the statistics of its sample individually. However, in the interface step, the calculation of mean and variance is based on the entire population data. This indicates that there is a discrepancy between the feature extraction procedure suggested by the batches and those by the inference net.

This difference does not cause much of a problem, if the batch size is relatively big, as that includes majority of the population, which tends to be more or less the same as the whole data. But when the batch size decreases, errors tend to predominate [13].

The mean and variance computed, use a very small data set, which indicate that the average value is not exactly accurate while incorporating a number of batches. Consider the case of the following dataset as an assumption:

{2,4,3,5,10,12,6,7,8,6,9,10}

The actual average value of the data:6.8 approximated as 7.

Clustering randomly towards a batch sizes of 2, and performing mean over the adjacent couples:

{3,4,11,6.5,7,9.5}.

Consider the case of a batch size 6. The means obtained are {6,7.6} The results obtained are closer in range to the actual average. The above example suggests that smaller batch sizes performs relatively poor in the process of sample mean and variance. Hence, implementation using lower GPUs, cannot take place efficiently through this algorithm.

Additionally, non- independent, identically trained mini batches perform poorly when fed into this situation [6]. Consider the case of a metric learning scenario. It is a usual trend to work on examples that are relatively similar. In a mini batch size of 36, it is a proposed trend to include 18 labels, with two examples in each. However, when this mixture is trained, it shifts towards the observed trend. It looks for a group of two similar elements and labels them accordingly. This biased feature extraction deteriorates when combined with other batches, thus decreasing its efficiency.

IV. OPTIMIZATION TECHNIQUE

4.1Batch Re-norm

Batch re-norm proposes two enhancements:

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 8, August 2017

- Every node's activation is computed with respect to its individual characteristics and hence, influence of the mini batch is avoided.

- The activations provided are regularized in a manner that commensurate the individual batch elements with the inference step.

In this method, Sergey Ioffe understood the need to decouple the dependence of each example on mini-batch [9]. A plausible alternative is the use of moving averages that reduce this dependency.
However, the process of computing an entire moving average during every batch cycle is computationally expensive; Also, such implementations can elicit undesirable results if gradient descent and the normalization step interact with each other in a manner that nullifies their effects.
Hence a modification in the existing batch normalization algorithm, is the inclusion of constants r, d which helps retain the characteristic of mini batch and equate the averages with inference mean through the following process:
Consider the case of a sample input x to a node, with mini batch statistics as μ_B, σ_B, and μ, σ be the moving average and standard deviation computed across the set of mini batches so far.

$$\frac{x_i-\mu}{\sigma}=\frac{x_i-\mu_B}{\sigma_B}*r+d, \quad \text{when} \quad r=\frac{\sigma_B}{\sigma},$$

$$d=\frac{\mu_B-\mu}{\sigma}.$$

This indicates that parameters r, d help translate a particular set of batch parameters to be in proportion with that of the inference step.They are treated as constants that are computed during the mini-batch training. These affine transformations are inserted across every normalized activation, and rectify the differences between the training and the end ensemble model. Thus, the activations now seem similar to the ones generated by inference, and hence can account for smaller batch sizes.

4.2 Algorithm

For every feature, across every batch the following steps are computed [9]:
Where x_i depicts input of each x from the mini-batch of size m, μ_B is the mean over the specific batch, σ_B^2 is the batch variance and y_i is the output corresponding to each input in batch. μ, σ as moving average and standard deviation computed across the set of population.

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

$$\sigma_B=\sqrt{\in+\frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2}$$

$$r=stop\_gradient\left(clip_{\left[\frac{1}{r\max},r\max\right]}\left(\frac{\sigma_B}{\sigma}\right)\right)$$

$$d=stop\_gradient\left(clip_{[-d\max,d\max]}\left(\frac{\mu_B-\mu}{\sigma}\right)\right)$$

$$\hat{x}_i=\frac{x_i-\mu_B}{\sigma_B}*r+d$$

The updation of the moving averages take place as:
$\mu=\mu+\alpha(\mu_B-\mu)$
$\sigma=\sigma+\alpha(\sigma_B-\sigma)$, where alpha is the learning rate parameter.

Finally, the inference step computes the output as:

$$y=\gamma\cdot\frac{x-\mu}{\sigma}+\beta$$

This concludes the batch re-norm algorithm.

4.3 Batch-renormalization effectiveness

Analysis of batch re-norm is based on its performance over a previously calculated primary model of [7] with training data of [8] and 50 synchronized workers [10] discerned through ImageNet validation data. The evaluation of batch re-normalization in this paper is based on the interpretation of results in the research work [9].

Training using smaller batches:

- The results produced 76.5% accuracy while training a sample of batch size of 4 with re-norm method, compared to the 74.2% accuracy produced with batch normalization.

- Although the accuracy improved, it still considered deficit to train a smaller batch size. Hence, it can be concluded that batch re-norm, improves training with mini batch sizes marginally, but does not eradicate the problem as a whole.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 8, August 2017

Training using non-i.i.d inputs through batches:

- The results obtained exhibit lower accuracy during the training range, hence calling for modifications in parameters. The two similar sets of counter parts in a batch are split into separate batches further.

- This ensures that a batch contains approximately one example per feature each, without having to look for two similar samples. The latter is a deviation from the concept of classification. This process produces an increased accuracy of 77.4% compared to batch norm whose value lingers around 67%.

- However, while performing the same steps through a higher batch size, the performance further improves providing an accuracy of 78%.

- Thus, batch re-norm enables better performance on non-i.i.d inputs when the batch size is comparatively big, and the existing method, though partially efficient, requires further enhancements to increase accuracy.

V. CONCLUSION

This paper provides a summary of the background behind batch normalization, its need and drawbacks. The in-depth analysis of its algorithm proved helpful towards fault detection and inefficiency criteria. Small batch size and non-independent parameters were unsuccessful in this classification.
Optimisation technique called batch-renormalisation was elaborated to understand the defects and enhance the network.
Further elaboration on batch-renormalisation was done to analyse its efficiency, which concluded that it cannot provide a complete cure for the problem.
Hence, this paper serves a brief summary for researches to explore the area of normalization, and come up with various other optimizations based on the records of previous findings cited.

REFERENCES

[1] TheSutskever, Ilya, Martens, James, Dahl, George E., and Hinton, Geoffrey E. On the importance of initialization and momentum in deep learning. In *ICML(3)*, volume 28 of *JMLR Proceedings*, pp. 1139–1147.JMLR.org, 2013.

[2] Shimodaira, Hidetoshi. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*,90(2):227–244, October 2000.

[3] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex,Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout:A simple way to prevent neural networks from overfitting.*J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[4] LeCun, Y., Bottou, L., Orr, G., and Muller, K. Efficient backprop. In Orr, G. and K., Muller (eds.), *Neural Networks:Tricks of the trade*. Springer, 1998b.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conferenceon Machine Learning (ICML-15)*, pages 448–456, 2015.

[6] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, 2004.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna.Rethinking the inception architecture for computer vision.In*Proceedings of the IEEE Conference on ComputerVision and Pattern Recognition*, pages 2818–2826, 2016.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh,S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein,A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[9] S. Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-

normalized models. arXiv preprint arXiv:1702.03275, 2017

[10] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[11] https: // gab41.lab41.org /batch-normalization-what the-hey- d480039a9e3b

[12] http://cs231n.github.io/optimization-1/ https :// www . semantic scholar .org/ paper / Image Net-pre- trained- models- with – batch – normalizati-Simon Rodner / 1d5fe8230 3712a7 0c1d231 ead2ee0 3f042d8ad70