Yet Another Batch Norm Analysis: Interim Report

G42 (s1837803, s1860947, s1751752)

Abstract

Batch normalization is a commonly used technique that allows deep neural networks to train for longer by normalizing layer inputs. While undeniably effective, the reason for this effectiveness has been under some debate. Several works have been published which investigate more into this deceptively simple method. Recently, it has been discovered that Batch Normalization has expressive power of its own and is able to achieve impressive results on deep enough networks. In this paper, we aim to investigate the consequences of this phenomenon, and it's limits. We will do this by training networks with everything but the BatchNorm parameters frozen, and by removing some of the BatchNorm layers to see how it affects the model. The networks were trained on the popular MNIST and CIFAR-10 datasets.

1. Introduction

Batch norm (BN) was initially introduced as a regularization tool to help alleviate the problem of Internal Covariate Shift (ICS), a phenomenon where the distrubution of a layer's inputs significantly change between training steps as a result of changes in earlier layers. Originally proposed by (Ioffe & Szegedy, 2015), it functions by introducing batch norm layers into the neural network architecture - these layers constrain their inputs to a normalised output distribution (normalised with respect to a given training batch), scaled and shifted according to learnable parameters γ and β.

BN is clearly a powerful tool, as it was shown by (Ioffe & Szegedy, 2015) to reduce the effects of the vanishing/exploding gradient problem (common issues in the training of deep networks, where the successive layers have a compounding shrinking/destabilising effect on gradients during backpropogation). BN also allows for higher learning rates and therefore enables faster network training, as well as improving robustness to various hyperparameter settings (Santurkar et al., 2018).

Since its original publication, the exact effect and theory behind BN have been debated. (Santurkar et al., 2018) suggest that BN does not reduce ICS (and indeed, that reducing ICS does not necessarily improve training), and instead claim its improvements are due to a smoother optimization landscape, and (Luo et al., 2018) highlight its role as a regularizer. It is clear that BN requires further study. In this paper we aim to investigate some of the basic assumptions made about BN. We hope to either increase confidence in BN or to discover potential improvements.

One aspect of BN which has not received a lot of attention is its inherent representitive power, first described by (Frankle et al., 2020). At the core of this ability are the scaling and shifting parameters (γ and β, introduced to prevent any loss of representative power occurring when BN constrains a layer's distribution 2.

As initial motivation, we ran a VGG-38 network on CIFAR-100, and plotted the average magnitude of gradients, showing that on average each BN parameter receives larger gradients and hence updates than the rest of the network (seen in 2).

2. Related Works

(Ioffe & Szegedy, 2015) Initially proposed BN. The paper argued for batch norm as a method for reducing ICS, and showed that it could lead to dramatically increased performance on ImageNet.

(Santurkar et al., 2018) investigated further the mechanism through which BN improves performance. The authors find that BNs reduction of ICS may not be strongly connected to its performance improvements, a claim they support showing that a BN network with random noise injected after BN layers at each time step (thus intentionally reintroducing ICS) has almost no difference in performance. They furthermore claim that BN may not even necessarily reduce ICS at all, and that BN's improvements in performance instead stem from smoothing the optimisation landscape.

(Frankle et al., 2020) Investigated the expressive power of the *gamma* and *beta* parameters. By training a model where all non-BN parameters were frozen to random values and achieving non-trivial performance, they showed that the BN parameters by themselves have learning power. The authors were able to achieve impressive accuracy on sufficiently deep networks. On a 866-deep ResNet the accuracy on CIFAR-10 is 82%, and a 200-deep network achieves 32% top-five accuracy on ImageNet, improved to 57% when the output linear layer was allowed to train as well.

Examining the parameters further, the authors discover that γ is very small in many cases, and if those values are clamped to zero it doesn't negatively affect the network. Here, the authors conclude that they "turn off" irrelevant random features. The papers find that values are smaller as networks become deeper, and hypothesize that they do

this to prevent exploding gradients. Gamma and beta make activations sparser. A key advantage of this architecture is that the that network takes up little space, and one only needs to the seed for the random values as well as the BatchNorm parameters.

3. Research Questions & Project Objectives

The objective of this paper is to investigate the consequences of the fact that BN's γ and β parameters have representational power of their own.

Specifically we aim to investigate if these parameters have the capacity to introduce overfitting, examine methods of reducing this overfitting should it occur. We also aim to identify techniques to further take advantage of the representational power of BN, should it be desirable.

We build upon previous investigations of BatchNorm, especially (Frankle et al., 2020) which identified the shifting and scaling parameters as a powerful learning tool - we are expanding on their work by explicitly looking at the consequences of this. We also wish to examine the effects of removing a certain amount of BatchNorm layers from various networks. If the model preforms comparatively well, that could mean improved training or inference times without sacrificing significant accuracy.

We hope, in more general terms, to contribute to the understanding of batch normalisation. Additionally we wish to refine it's usage, as we hypothesize that adding a BatchNorm layer may not always be needed.

4. Dataset & task

MNIST is a dataset of greyscale handwritten digits with 60.000 training and and 10.000 test examples. The images have been normalized, anti-aliased and are 28x28 pixels. It was used by the original Batch Normalization paper.

CIFAR-10 and CIFAR-100 (figure 1) consist of coloured images divided into 10 and 100 classes respectively, with 50,000 training examples and 10,000 test examples. It was used by (Frankle et al., 2020). As the time of writing, MNIST and CIFAR-10 has been used over 5000 and 8000 times respectively, according to Papers With Code.

Both of these datasets present a computer vision classification task - identifying handwitten digits in the case of MNIST and photographed objects in the case of CIFAR. We use this specific type of task, as there is a broad body of literature focused on using BN for computer vision classification

We want to investigate overfitting when learning with BatchNorm, so we should record both classification accuracy and generalisation gap.

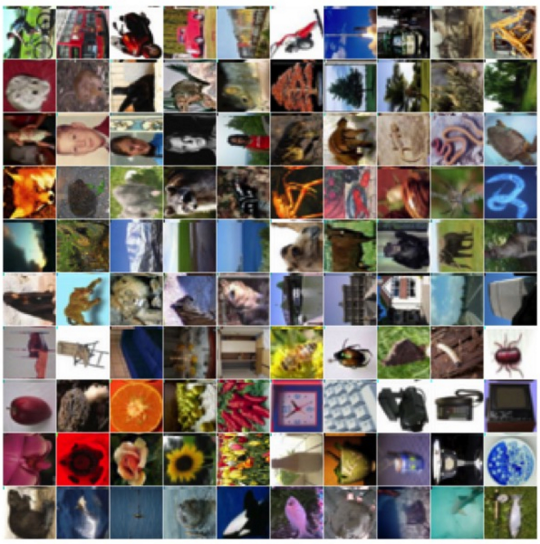

Figure 1. Sample images from the CIFAR-100 dataset.

5. Methodology

All networks apart from VGG-38 are trained with the same schedule as (Frankle et al., 2020):

- Data augmentation by normalizing per-pixel, randomly flipping horizontally, and randomly translating by up to four pixels in each direction. We use standard weight decay of 1e-4 on all parameters, including BatchNorm.
- We initialize all networks using He normal initialization (He et al., 2015b)
- The γ parameters of BatchNorm are sampled uniformly from [0, 1] and the β parameters are set to 0.
- We train until convergence using SGD with momentum (0.9) and a batch size of 128.
- The initial learning rate is 0.1 and drops by 10x at epochs 80 and 120.

Due to the requirement for very deep networks we use Residual Networks (He et al., 2016) as a baseline and backbone for all experiments. Residual networks introduce skip connections between every other layer, which make it easier to learn identity mappings between blocks of layers. This not only alleviates the vanishing gradients problem by allowing gradients to flow uninterrupted to earlier layers, but also means extensions to existing networks do not degrade performance since the network can learn to ignore them.

For small parts of the analysis we use VGG-38 (Simonyan & Zisserman, 2015) with additional residual connections, because of its regular architecture and simplicity.

6. Experiments

For our initial experiments we wanted to examine the relative importance of BatchNorm's γ and β when compared to other parameters in the networks. Therefore, we ran a VGG38 network with residual connections for 100 epochs. We chose this architecture because as we had experience with it from previous work.

We intend to run the following experiments to confirm BNs expressive power, to investigate the consequences of that power, and to find methods to improve training with those consequences in mind:

- Standard ResNets with 50, 152, and 200 convolutional layers, with and without BN. These experiments will provide a baseline to which we can compare later experiments.
- ResNets with 50, 152, and 200 convolutional layers with everything frozen but batchnorm layers. We intend to confirm (Frankle et al., 2020)'s results, and to specifically investigate how BNs learning power changes with network depth.
- ResNets with 50, 152, and 200 convolutional layers with "changed" batchnorm (e.g regularised BN parameters, differing distributions, removed shifting and scaling on some layers, or other modifications). We will investigate if there is overfitting that can be reduced, if we can improve upon BNs learning power, etc.

We also intend to run some experiments to investigate how BN interacts with learning when not all layers have BN layers after them.

- Standard ResNets with 50, 152, and 200 convolutional layers, as well as BN layers. This will provide baseline values for the next experiments.
- Standard ResNets with 50, 152, and 200 convolution layers, with a reduced number of batch norm layers (i.e after every 2 conv layers, after every 5 conv layers). We aim to find out if the expressive power of larger 'blocks' in the architecture can be improved by not regularising internally.
- Standard ResNets with 25, 76, and 100 convolutional layers, with full BN. This would be to confirm that the large 'blocks' in the previous experiments where contributing to expressive power, and and changes did not simply come from a lower number of BN layers.

7. Interim conclusions

In the limited initial experiments we ave confirmed out initial intuitions about BatchNorm, namely that its parameters are on average more important than the convolutional ones. Figure 2 shows that over 100 epochs, the absolute gradient of the BatchNorm layers is far higher than the rest and stays high. This means that these layers keep learning far more than the other ones. However, the core question is if this learning is beneficial.

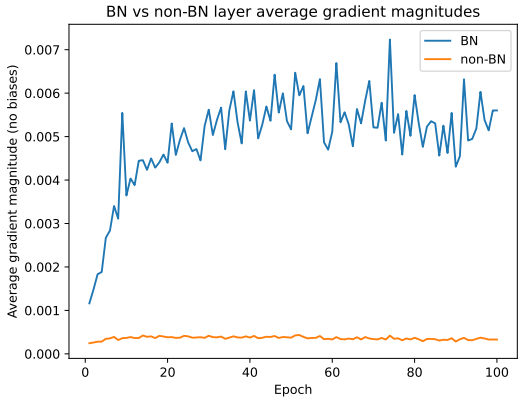

Figure 2. Comparison of average gradient magnitudes in VGG-38 network

8. Plan

For the rest of this project we intend to run the experiments defined above within two weeks, and then analyze the results. Then we would potentially define more experiments based on those results. If we can quickly achieve interesting results we can quite easily define more experiments that use BatchNorm in different ways. The risks involved would be that the experiments would yield little interesting data for analysis, or that if we find that we do not identify any solutions to any issues we find with BatchNorm. As the project involves somewhat advanced data, there is also a risk of us misinterpreting the data. Therefore we will make sure that our experiments are easy to reproduce.
A backup plan if we only find negative results, is to spend the rest of the project analyzing the data and try to extract some meaningful conclusions from it.

References

Frankle, Jonathan, Schwab, David J, and Morcos, Ari S. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. *arXiv preprint arXiv:2003.00152*, 2020.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Luo, Ping, Wang, Xinjiang, Shao, Wenqi, and Peng, Zhanglin. Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*, 2018.

Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, and Mądry, Aleksander. How does batch normalization help optimization? In *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488–2498, 2018.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition, 2015.