MLP Coursework 2: Investigating the Optimization of Convolutional Networks

s1705386

Abstract

The vanishing gradients problem (VGP) has been one of the grand obstacles hindering the trainability of deep neural networks (DNNs). One technique which largely alleviates this problem is that of Batch Normalization (BatchNorm) which has since seen wide-spread use in convolutional neural network (CNN) architectures yet, despite it's popularity, fundamental reasons for BatchNorm's effectiveness remain elusive. In this work we train a VGG38 CNN on CIFAR100 and demonstrate that BatchNorm has a smoothing effect on the optimization landscape that is further enhanced by the use of larger batch sizes.

1. Introduction

Over the past decade deep learning has been at the spearhead for tackling problems in the field of computer science such as image classification (Krizhevsky et al., 2012; He et al., 2016; Huang et al., 2017) and speech recognition (Graves et al., 2013).

Convolutional neural networks (CNNs) are one such type of a deep learning algorithm that have been integral to advancing the state-of-the-art in the field of computer vision. Through their architecture of sequentially stacked convolutional layers, deep CNNs are naturally capable of forming high level abstractions of complicated environmental features, and hence have an astounding capacity for spatially invariant feature extraction in complex image datasets. It is with depth comes a greater capacity for learning and a trend of greater generalization performance (Simonyan & Zisserman, 2015; Krizhevsky et al., 2012; Huang et al., 2016).

Deeper networks, however, come at the cost of being significantly harder to train (Bengio et al., 1994). Perhaps the most notorious obstacle hindering this training is the vanishing (and exploding) gradients problem (VGP) that has plagued the stochastic gradient descent optimizers we depend on so heavily. It is thus imperative this problem is addressed if we wish to continue to make advancements in the field of deep learning.

A number of practical solutions have been proposed for tackling the VGP over the years including the introduction of novel activation functions (Dahl et al., 2013), careful initialization schemes (Glorot & Bengio, 2010), and architecture modifications such as batch normalization, residual networks and dense networks (Ioffe & Szegedy, 2015; He et al., 2016; Huang et al., 2017).

Batch normalization (BatchNorm), in particular, has been incredibly effective in alleviating the VGP yet still little is understood about the reasons for its effectiveness. This observation prompted the following research question:

How does BatchNorm solve the VGP?

Our report is organized as follows. In §2, we explore the adversaries faced when training deep CNNs. Then, in §3, we conduct a literature review on some of the more prominent solutions to the VGP including BatchNorm. We break down the inner workings of BatchNorm in §4 before conducting our experiments and analysis into the posed research question in §5. Further related works and possible advancements are discussed in §6 and §7.

2. Identifying training problems of a deep CNN

At the heart of the learning process for all deep neural networks (DNNs) is the backpropagation algorithm (Rumelhart et al., 1986). Stochastic gradient descent (SGD) optimizers are used to realise this algorithm, with the objective of minimizing a high-dimensional non-convex loss function (Du et al., 2019). This process is known as training.

During SGD, we divide our training set of size N into B mini-batches which are randomly passed to the network as training data. A single SGD update is then defined to be: $\theta \leftarrow \theta - \frac{\eta}{B}\sum_{b=1}^{B}\frac{\partial \mathcal{L}}{\partial \theta}$ where η is the learning rate hyperparameter, and $\frac{\partial \mathcal{L}}{\partial \theta}$ is the gradient of the loss function with respect to the parameter of interest. The gradients can be interpreted as the *learning signal* which determine the direction of travel in the optimization landscape and are hence vital to the learning process.

One of the most prominent obstacles in training DNNs is the VGP (Bengio et al., 1994) where suppose one has a DNN with a total of L fully-connected layers. Let $\mathcal{L}$ denote our loss function measuring the performance of our network on some given task, and let $\mathbf{h}^{(\ell)}$ denote the hidden state of the network at layer ℓ. For this illustrative example, let us define forward propagation as $\mathbf{h}^{(\ell)} = g\left(W^{(\ell)}\mathbf{h}^{(\ell-\mathbf{1})}\right)$ where the biases have been omitted for clarity, and g is an activation function. To obtain the gradient of our loss function with respect to the first hidden state, $\mathbf{h}^{(1)}$, the chain rule is repeatedly applied to give the following expression:

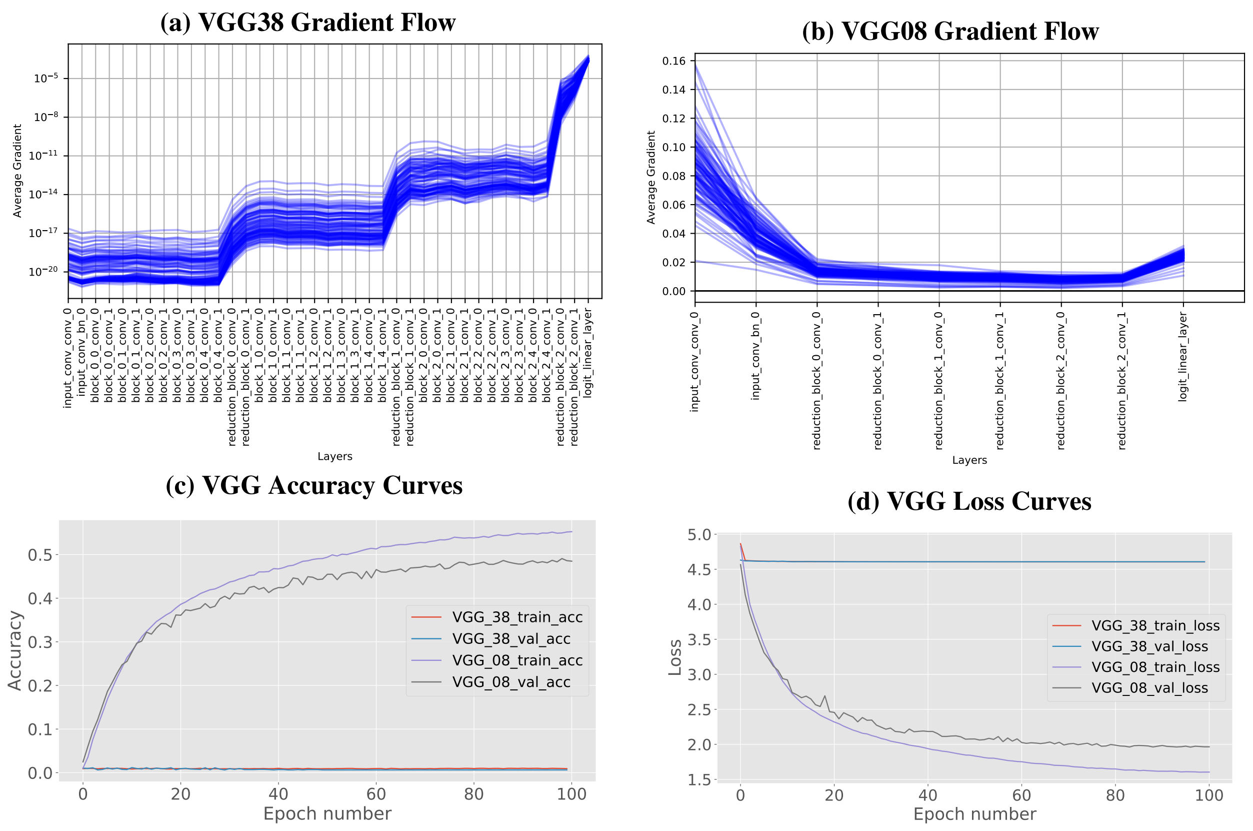

Figure 1. Panels (a) and (b) present the per layer average mean absolute value of the gradient of the loss function for the VGG38 and VGG08 models respectively. Panels (c) and (d) depict the corresponding accuracy and loss curves (respectively) of both of the aforementioned models.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(L)}} \prod_{\ell=1}^{L-1} \frac{\partial \mathbf{h}^{(\ell+1)}}{\partial \mathbf{h}^{(\ell)}} \tag{1}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(L)}} \prod_{\ell=1}^{L-1} W^{(\ell)} g'\left(W^{(\ell)} \mathbf{h}^{(\ell-1)}\right) \tag{2}$$

Now, for very deep networks where L is large, this repeated application of the chain rule becomes detrimental to learning. Pascanu *et al.* proved that if the largest eigenvalue of every parameter matrix, $W^{(\ell)}$, is less than 1, the gradients will shrink exponentially (Pascanu et al., 2013). It is this mechanism that is central to the VGP. An analogous argument can be made for the exploding gradients problem where the eigenvalues of $W^{(\ell)}$ are greater than 1.

Furthermore, it has been empirically observed that when neural networks become sufficiently deep, loss function landscapes abruptly transition from being nearly convex, to highly chaotic with many spurious local minima (Li et al., 2018). Hence, upon initialization of our model parameters, our gradient based optimizer can fall prey to abrupt changes in the loss landscape or become stuck on an inescapable flat plane. This relationship between network depth and anarchic loss landscapes thus greatly reduces our ability to globally optimize the loss function and hence train the network.

Figures 1(a) and 1(b) depict the gradient flows through a VGG08 and VGG38 (Simonyan & Zisserman, 2015) CNN trained and evaluated for a total of 100 epochs on the CIFAR100 dataset. During the training process, we see a stark contrast in the gradient flows through each layer of the networks, with VGG38 gradients being several order of magnitude smaller than those of VGG08. The VGG38 gradients are so small, in fact, that we observe in Figures 1(c) and 1(d) that this network even fails to train achieving $\approx 1\%$ accuracy on the training set. We can hence confidently diagnose VGG38 to be suffering from the VGP which is further justified via the diminishing gradient magnitudes as the input layer is approached. This is congruent with aforementioned argument and Equations (1) and (2).

3. Background Literature

In this section we will highlight some of the most influential papers that have been central to overcoming the VGP in deep CNNs.

Residual Networks (He et al., 2016). One interpretation of how the VGP arises is that as non-linear layers are continuously stacked between between the input and output of your network, the connection between these variables grows increasingly weaker. This results in the gradients becoming increasingly scrambled as they are propagated back through the network and the desired mapping between input and output being lost. The authors observed this manifest through a deeper 56-layer "plain" neural network counter-intuitively achieve a higher training error than a more-shallow, 20-layer "plain" network. Residual networks, colloquially known as ResNets, aim to alleviate this dispute through the incorporation of *skip connections* (Bishop, 2014) into the network architecture. Therefore, instead of hoping a block of stacked layers directly fit a desired underlying mapping, skip connections explicitly enable layers to additionally fit a *residual* or *identity* mapping. For example, let $\mathcal{H}(\mathbf{x})$ denote the desired underlying mapping of a block of CNN layers where $\mathbf{x}$ denotes the input to this convolutional block. Through the addition of skip connections, this block is instructed to fit an alternative mapping, $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$, hence recasting the original as $\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$. The authors argue that this new mapping is significantly easier to optimize since if an identity mapping were optimal, the network could comfortably learn to push the residual to zero rather than attempting to fit an identity mapping via a stack of nonlinear layers. They bolster their argument by successfully training ResNets with depths exceeding 1000 layers on the CIFAR10 dataset (Krizhevsky, 2009). Prior to their work, training even a 100-layer was perceived as a great challenge within the deep learning community. The addition of skip connections solves the VGP through enabling information to flow more freely throughout the network architecture without the addition of neither extra parameters, nor computational complexity.

Densely Connected Networks (Huang et al., 2017). DenseNet architectures build upon the merits of ResNets and hence employ a similar strategy to alleviating the VGP. Rather than adding the original input to the output of a convolutional layer, these networks preserve the original features directly by concatenation. This has the effect of each subsequent layer taking all preceding learnt feature-maps as input thus introducing a total $\frac{L(L+1)}{2}$ connections to a deep CNN with L layers. Mathematically, this can be described as:

$$\mathbf{x}_\ell = \mathcal{H}_\ell\left([\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{(\ell-1)}]\right) \tag{3}$$

where $[\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{(\ell-1)}]$ refers to the concatenation of the $\ell - 1$ feature maps from the previous layers. This provides maximal information flow through the network and hence increases the ease with which gradients propagate. Recent work has shown that many layers provide diminishing returns to the performance of a network and can in fact be safely dropped at training time (Huang et al., 2016) to prevent the learning of redundant feature maps. Hence by shedding layers in favour of connectivity, DenseNets achieve greater performance than ResNets with substantially fewer parameters/layers.

Batch Normalization (Ioffe & Szegedy, 2015). Unlike the aforementioned techniques, Batch Normalization (BatchNorm) sought to solve the VGP through addressing the related problem of *internal covariate shift* (ICS). The authors argue that without batch normalization, the distribution of each layer's inputs can vary significantly due to the stochastic nature of randomly sampling mini-batches from your training set. Layers in the network hence must continuously adapt to these high variance distributions which hinders the rate of convergence of our SGD optimizer. This optimization problem is exacerbated further with network depth due to the updating of parameters at layer ℓ being dependent on the previous $\ell - 1$ layers.

It is therefore instructive to embed the normalization of training data into the network architecture after work from LeCun *et al.* showed that training converges faster with this addition (LeCun et al., 2012). Through standardizing the inputs to each layer, we take a step towards achieving the fixed distributions of inputs that remove the ill effects of ICS. Ioffe and Szegedy demonstrate the effectiveness of their technique through training an ensemble of BatchNorm networks which achieve an accuracy on the ImageNet Classification task exceeding that of humans in 14 times fewer training steps than the state-of-the-art of the time.

It should be noted, however, that the exact reasons for BatchNorm's proficiency are still poorly understood, and that recent work has shown that the integration of distributional stability of layer inputs has little to do with its success (Santurkar et al., 2018). Santurkar *et al.* present that BatchNorm actually has a *smoothing* effect on the optimization landscape thus enabling a more predictive and stable behavior of the gradients and allowing for faster training. This notion is congruent with work conducted by Li *et al.* who independently show that both ResNets and DenseNets also have a smoothing effect on the optimization landscape (Li et al., 2018). The union of these three techniques thus leads us to conclude that smoothing the optimization landscape is the most fundamental component in alleviating the VGP.

4. Solution Overview

Since its publication in 2015, BatchNorm has become a staple in the diet of deep learning practitioners and researchers, and was even implemented in both the ResNet and DenseNet paper previously outlined. Given this technique's simplicity, prevalence, and controversy, in this report we choose to further elucidate how this per-layer normalization technique alleviates the VGP.

As mentioned in §3, BatchNorm (BN) is per-layer transformation that is performed to *whiten* the activations originating from each layer. Unfortunately, a full standardization of each layer's inputs would be too costly during training, hence the BN transformation is performed on each scalar feature independently, for example, for a d-dimensional input $\mathbf{x} = (x^{(1)}, ..., x^{(d)})$,

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}; \quad \forall\, k = 1, ..., \mathrm{d}. \tag{4}$$

where the expectation and variance are computed over the training set. Note that this normalization step will suppress the representational capabilities of the layer prompting the addition of a pair of *learned* re-scaling and re-shifting parameters, $\gamma^{(k)}$ and $\beta^{(k)}$, to ensure the identity mapping remains attainable:

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}; \quad \forall\, k = 1, ..., d. \tag{5}$$

Hence, for a mini-batch of size b defined as $\mathcal{B} = \{x^{(k)}_{1...b}\}$, we denote the *Batch Normalizing Transform* as

$$\text{BN}_{\gamma,\beta} : x^{(k)}_{1...b} \longrightarrow y^{(k)}_{1...\text{b}}; \quad \forall\, k = 1, ..., \text{d}. \tag{6}$$

We summarize the BN transformation in Algorithm 1 below where the addition of ϵ is purely for numerical stability purposes.

Algorithm 1 Batch Normalization Transform, applied to activation x over a mini-batch. Let the index, $x^{(k)}$ be omitted for clarity.

```
Input: Values of $x$ over a mini-batch $\mathcal{B} = \{x_{1...b}\}$
       Parameters to be learned: $\gamma, \beta$
Output: Batch-normalised inputs $y_i$

$\mu_{\mathcal{B}} \longleftarrow \frac{1}{b}\sum_{i=1}^{b} x_i$ {mini-batch mean}
$\sigma^2_{\mathcal{B}} \longleftarrow \frac{1}{b}\sum_{i=1}^{b}(x_i - \mu_{\mathcal{B}})^2$ {mini-batch variance}
$\hat{x}_i \longleftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$ {normalize}
$y_i \longleftarrow \gamma\hat{x}_i + \beta$ {scale and shift}
```

The efficacy of BatchNorm is thought to originate from the normalization of activations which in turn addresses the adverse effects of ICS. Ioffe and Szegedy argue that each BatchNorm layer can be perceived as a sub-network with fixed mean and variance (thanks to the normalization step). Hence, during the training of VGG38, although the joint distribution of activation inputs across a mini-batch may vary over the course of training, the training of these sub-networks is accelerated which consequently accelerates the training of the network as a whole.

This argument is intuitively logical, however, recent work has shown that BatchNorm has a the more fundamental benefit of smoothing the optimization landscape during training (Santurkar et al., 2018) thus enhancing the predictive power of gradients as our guide to the global minimum. Furthermore, a smoother optimization landscape should additionally enable the use of a wider range of learning rates and initialization schemes which is congruent with the findings of Ioffe and Szegedy in the original BatchNorm paper (Ioffe & Szegedy, 2015).

5. Experiments

In this section, we investigate the mechanism with which BatchNorm solves the VGP, specifically, does BatchNorm improve the Lipschitzness continuity of the loss function i.e. make the loss landscape smoother? An *improved* Lipschitzness is to be interpreted as the loss, and gradient magnitudes, changing at a smaller rate and is encapsulated by the following mathematical statement $|f(x_1) - f(x_2)| < L||x_1 - x_2||$ for all x_1 and x_2 where L is a positive, real constant.

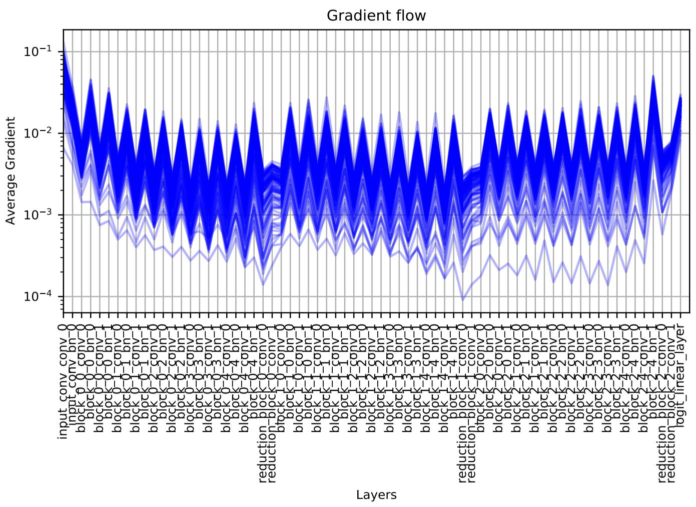

Figure 2. We present the per layer average mean absolute value of the gradient of the loss function (gradient flow) for a VGG38 CNN with Batch Normalization implemented at every convolutional layer. The CNN was trained with a batch size of 100 and a default Adam optimizer (learning rate of 0.001) with cosine annealing.

5.1. CIFAR100

All following experiments were conducted on the CIFAR100 (Krizhevsky, 2009) classification task; a common benchmark in the field of computer vision. This dataset consists of $60,000$, 32×32 pixel colour images belonging to 100 classes, with 600 images per class. The images were partitioned into a training, validation and test set with a total of 47,500, 2500, and 10,000 images in each set respectively. Finally, since it has been shown L2 regularization has little to no actual regularizing effect when combined with BatchNorm, (van Laarhoven, 2017) we instead impose random cropping and horizontal flips to augment the CIFAR100 dataset (Shorten & Khoshgoftaar, 2019) to prevent overfitting.

5.2. VGG38 Architecture and Baselines

The structure of the VGG38 architecture that previously failed to learn (see Figure 1) can be summarised as: InputBlock, Stage1, Stage2, Stage3, and FullyConnected. The InputBlock takes in a mini-batch of images before passing them through a 2D convolution (Conv2D) with a kernel size of 3, padding of 1 and a total of 32 filters. We note that all following Conv2D transformations also use these hyperparameters. These outputs are then passed through a LeakyReLU activation function with a negative slope of 0.01. Following the input block, the activations are processed through a series of *stages* which consist of a set of convolutional and dimensionality reduction blocks. Each stage consists of 5 sequential convolutional blocks before terminating with a dimensionality reduction block. We define a convolutional block to have the following transformation procedure - Conv2D, LeakyReLU, Conv2D,

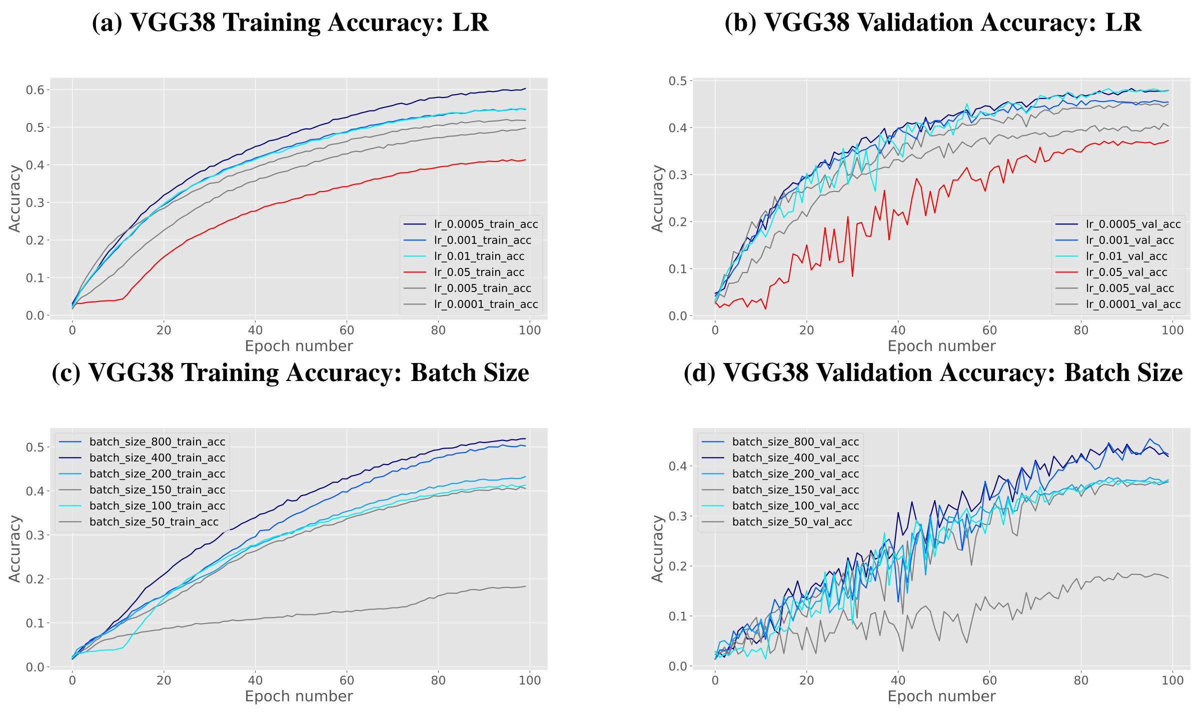

Figure 3. Panels (a) and (b) present the per layer average mean absolute value of the gradient of the loss function for the VGG38 and VGG08 models respectively. Panels (c) and (d) depict the corresponding accuracy and loss curves (respectively) of both of the aforementioned models.

LeakyReLU - whilst a dimensionality reduction block is defined analogously with an AveragePooling transformation proceeding the first LeakyReLU activation. AveragePooling differs from Conv2D in that the kernel is now of size 2 which has a desireable dimensionality reduction effect of halving the dimensions of our feature maps. The VGG38 architecture concludes with a FullyConnected layer consisting of 100 hidden units (1 for each class in the CIFAR100 dataset) and softmax activation functions to yield the class predictions as a vector of probabilities.

This architecture was trained for a total of 100 epochs using an Adam SGD optimizer (Kingma & Ba, 2015) with cosine annealing (without warm restarts, $\eta_{min} = 0.00002$, $T_{max} = 100$) (Loshchilov & Hutter, 2017) and a batch size of 100 images. Cross-entropy loss was chosen to measure the accuracy of predictions made by the CNN. The results of this training can be observed in Panels (a), (c) and (d) of Figure 1 in §2 where the architecture suffers greatly from the VGP and fails to learn.

5.3. Batch Normalization

The baseline VGG38 architecture was modified through the addition of a BatchNorm (BN) layer proceeding every Conv2D transformation in each convolutional block. As per the analytical and empirical results outlined in §3 and §4, one would expect far richer gradient flows, and hence passage of information, through the VGG38 architecture.

Our intuitions are confirmed by the behaviour observed in Figure 2 (above) where the gradient magnitudes remain firmly within the range of $[10^{-4}, 10^{-1}]$; an ample range for effective learning to take place. This is a dramatic improvement to what was observed in Figure 1(a) in §2, where gradient magnitudes diminished to values as low as 10^{-20} preventing any meaningful SGD updates.

5.4. Lipschitz Smoothing

In the context of our baseline experiment, it is clear that BatchNorm has completely eradicated the problem of vanishing gradients. We hypothesized, in accordance with work conducted by Santurkar *et al.* (Santurkar et al., 2018), that BatchNorm has a rather fundamental effect on the training process in smoothing the optimization landscape. If this were true, one would expect to be able to successfully train with larger learning rates, η, since larger steps can be taken by an Adam optimizer without falling subject to the perils of spurious local minima or considerable flat regions in the loss landscape. It should be noted that without BatchNorm, no training was even possible with the VGG38 architecture thus the gradient flow plot presented in Figure 2 is already evidence of some loss landscape smoothing occurring. To test this hypothesis further, we re-trained VGG38 whilst varying learning rates from 0.0001 to 0.05, rather than using the Adam default of $\eta = 0.001$, with the expectation of greater performance from larger learning rates.

Our experimental results are presented in panels (a) and (b) of Figure 3 (above) which depict the training and validation accuracy of the VGG38 CNN with BatchNorm imposed (more details shown in Table 1).

Contrary to our previously outlined expectations, the greatest training performance is observed for when the learning rate that is *half* the original Adam default, $\eta = 0.0005$. However, we additionally observe practically equal training performance for learning rates of 0.001 (Adam default) and 0.01 (a 10x increase). Whilst these learning rates are equals on the training set, it is revealed in panel 3(b) and Table 1 that the larger learning rate of 0.01 has enhanced generalisation performance achieving an accuracy of 47.84% on the validation set whilst the baseline learning rate of 0.001 only manages 45.44% when evaluated at epoch 100.

Learning Rate	Loss Train	Loss Valid	Acc Train	Acc Valid
0.0001	1.815(3)	2.311(1)	0.493(2)	0.402(1)
0.0005	**1.371(2)**	1.953(1)	**0.599(2)**	0.477(1)
0.001	1.601(1)	1.974(2)	0.547(2)	0.455(1)
0.005	1.737(1)	2.056(3)	0.518(1)	0.448(1)
0.01	1.603(3)	**1.943(1)**	0.547(3)	**0.479(1)**
0.05	2.096(3)	2.331(1)	0.411(2)	0.367(2)

Table 1. Training and validation accuracy and loss for a set of a learning rates averaged over the final 5 epochs. The notation 0.479(4) is to be interpreted as 0.479 ± 0.004

.

The previously outlined results were taken to be inconclusive regarding how BatchNorm affects the optimization landscape and hence further experiments were conducted but now adjusting the batch size used at training time. The learning rate was fixed in accordance with the **worst** performing configuration, $\eta = 0.05$, which has been highlighted in red in Figure 3 below. Batch sizes were varied and took on values in the range of $[50, 800]$.

It should be noted that the gradient of the loss over a mini-batch is an *estimate* of the gradient over the entire training set. One should thus expect the quality of this estimate to improve as the batch size is increased (Ioffe, 2017). We observe this expected behaviour in panels (c) and (d) of Figure 3 (more details can also be observed in Table 2) where a batch size of 400 achieved a training accuracy of 51.88% and a validation accuracy of 41.89% evaluated at the 100th epoch relative to the baseline model (batch size of 100) achieving only 41.33% and 37.24% on those metrics respectively.

Batch Size	Val Acc
50	0.180(1)
100	0.368(1)
150	0.365(2)
200	0.372(1)
400	0.427(2)
800	**0.438(2)**

Table 2. Dependence on the validation accuracy of VGG38 models on batch size a learning rate of $\eta = 0.05$. The notation 0.479(4) is to be interpreted as 0.479 ± 0.004. The results presented are the average over the final 5 epochs.

At the more fundamental level of the loss landscape, these results are indicative of the batch size variable having a significant effect in improving the Lipschitzness of the loss landscape with Batch Normalization implemented.

6. Discussion

In summary, the results depicted in Figure 2 clearly demonstrate that Batch Normalization is extremely effective in mitigating the unwanted consequences of the VGP. Figure 3 builds on this conclusion and attempts to elucidate the underlying mechanism for BatchNorm's effectiveness. In line with results first observed by Ioffe and Szegedy, we observe the training process of a VGG38 CNN have increased robustness to a wider range of learning rates (Ioffe & Szegedy, 2015). Following the lead of Santurkar *et al.*, we interpret this result as BatchNorm having a smoothing effect on the loss landscape which is formally denoted as improving it's Lipschitzness (Santurkar et al., 2018).

To consolidate these findings, we additionally observe in further experiments a significant increase in the training and validation accuracy for a VGG38 CNN with learning rate of 0.05 by larger batch sizes of 400 and 800 relative to a batch size of 100 baseline. This result is suggestive that a larger batch size can result is enhancement in the smoothing effects of BatchNorm on the loss landscape.

However, this interpretation cannot be taken to the extremes and more work is yet to be done. Whilst we gain greater predictive performance from the BatchNorm estimators for large batch sizes, a point is eventually reached where the batch size is so large that the benefits stowed upon us by through SGD are lost (Kleinberg et al., 2018). There is hence a delicate balance to be struck for batch sizes when BatchNorm has been implemented which requires further investigation.

7. Conclusions

In this report we have explored the general adversities faced when optimizing the loss landscapes of deep convolutional neural networks for training, specifically, the VGP. Through analysis of the relevant literature and careful experimentation, we reveal that Batch Normalization has a significant effect in alleviating the VGP through smoothing the optimization landscape. This effect was observed to be more profound for larger batch sizes of 400 and 800 compared to a baseline of 100.

Whilst the results of our experiments are suggestive of batch normalization smoothing the loss landscape, they are far from conclusive. Akin to work conducted by Li *et al.*, future work should look into actually visualising the loss landscape of a deep CNN with, and without, Batch Normalization implemented (Li et al., 2018). Furthermore, through this work, one could also obtain an intuitive visual understanding of how batch size affects the smoothing capabilities of the BatchNorm transform. We believe the results produced by this potential work would be far more interpretable and probe deeper into the mechanism that makes BatchNorm so effective in training deep CNNs.

References

Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2), 1994. ISSN 19410093. doi: 10.1109/72.279181.

Bishop, Christopher M. Bishop - Pattern Recognition And Machine Learning - Springer 2006. *Antimicrobial agents and chemotherapy*, 58(12), 2014. ISSN 1098-6596.

Dahl, George E., Sainath, Tara N., and Hinton, Geoffrey E. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013. doi: 10.1109/ICASSP.2013.6639346.

Du, Simon S., Lee, Jason D., Li, Haochuan, Wang, Liwei, and Zhai, Xiyu. Gradient descent finds global minima of deep neural networks. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, 2019.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Journal of Machine Learning Research*, volume 9, 2010.

Graves, Alex, Mohamed, Abdel Rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013. doi: 10.1109/ICASSP.2013.6638947.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, 2016. doi: 10.1109/CVPR.2016.90.

Huang, Gao, Sun, Yu, Liu, Zhuang, Sedra, Daniel, and Weinberger, Kilian Q. Deep networks with stochastic depth. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9908 LNCS, 2016. doi: 10.1007/978-3-319-46493-0{_}39.

Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens, and Weinberger, Kilian Q. Densely connected convolutional networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January, 2017. doi: 10.1109/CVPR.2017.243.

Ioffe, Sergey. Batch Renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in Neural Information Processing Systems*, volume 2017-December, 2017.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, 2015.

Kingma, Diederik P. and Ba, Jimmy Lei. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

Kleinberg, Robert, Li, Yuanzhi, and Yuan, Yang. An alternative view: When does SGD escape local minima? In *35th International Conference on Machine Learning, ICML 2018*, volume 6, 2018.

Krizhevsky, Alex. Learning Multiple Layers of Features from Tiny Images. *... Science Department, University of Toronto, Tech. ...*, 2009. ISSN 1098-6596. doi: 10.1.1.222.9220.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 2, 2012. doi: 10.1061/(ASCE)GT.1943-5606.0001284.

LeCun, Yann A., Bottou, Léon, Orr, Genevieve B., and Müller, Klaus-Robert. Efficient BackProp. 2012. doi: 10.1007/978-3-642-35289-8{_}3.

Li, Hao, Xu, Zheng, Taylor, Gavin, Studer, Christoph, and Goldstein, Tom. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, volume 2018-December, 2018.

Loshchilov, Ilya and Hutter, Frank. SGDR: Stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. In *30th International Conference on Machine Learning, ICML 2013*, number PART 3, 2013.

Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. Learning representations by back-propagating errors. *Nature*, 323(6088), 1986. ISSN 00280836. doi: 10.1038/323533a0.

Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, and Madry, Aleksander. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, volume 2018-December, 2018.

Shorten, Connor and Khoshgoftaar, Taghi M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), 2019. ISSN 21961115. doi: 10.1186/s40537-019-0197-0.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

van Laarhoven, Twan. L2 Regularization versus Batch and Weight Normalization. *CoRR*, abs/1706.05350, 2017. URL http://arxiv.org/abs/1706.05350.