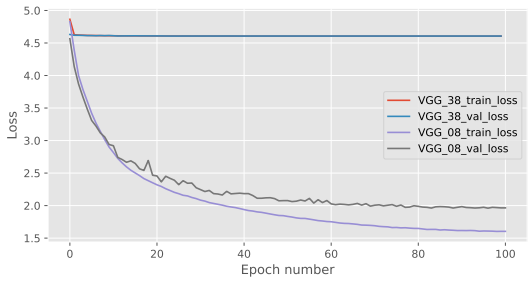
MLP Coursework 2

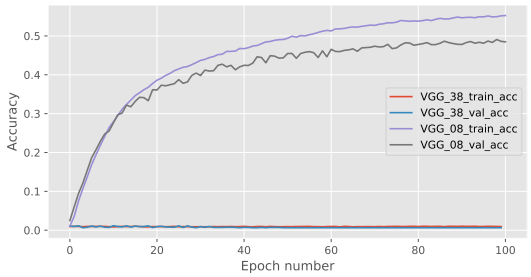
s2101367

Abstract

Deep neural networks have become the state-of-the-art in many standard computer vision problems thanks to their powerful representations and availability of large labeled datasets. While very deep networks allow for learning more levels of abstractions in their layers from the data, training these models successfully is a challenging task due to problematic gradient flow through the layers, known as vanishing/exploding gradient problem. In this report, we first analyze this problem in VGG models with 8 and 38 hidden layers on the CIFAR100 image dataset, by monitoring the gradient flow during training. We explore known solutions to this problem including batch normalization or residual connections, and explain their theory and implementation details. Our experiments show that batch normalization and residual connections effectively address the aforementioned problem and hence enable a deeper model to outperform shallower ones in the same experimental setup.



(a) Cross entropy error per epoch



(b) Classification accuracy per epoch

Figure 1. Training curves for VGG08 and VGG38 in terms of (a) cross-entropy error and (b) classification accuracy

1. Introduction

Despite the remarkable progress of modern convolutional neural networks (CNNs) in image classification problems (Simonyan & Zisserman, 2014; He et al., 2016), training very deep networks is a challenging procedure. One of the major problems is the Vanishing Gradient Problem (VGP), a phenomenon where the gradients of the error function with respect to network weights shrink to zero, as they backpropagate to earlier layers, hence preventing effective weight updates. This phenomenon is prevalent and has been extensively studied in various deep neural networks including feedforward networks (Glorot & Bengio, 2010), RNNs (Bengio et al., 1993), and CNNs (He et al., 2016). Multiple solutions have been proposed to mitigate this problem by using weight initialization strategies (Glorot & Bengio, 2010), activation functions (Glorot & Bengio, 2010), input normalization (Bishop et al., 1995), batch normalization (Ioffe & Szegedy, 2015), and shortcut connections (He et al., 2016; Huang et al., 2017).

This report focuses on diagnosing the VGP occurring in the VGG38 model[1] and addressing it by implementing two standard solutions. In particular, we first study a "broken" network in terms of its gradient flow, L1 norm of gradients with respect to its weights for each layer and contrast it to ones in the healthy and shallower VGG08 to pinpoint the problem. Next, we review two standard solutions for this problem, batch normalization (BN) (Ioffe & Szegedy, 2015) and residual connections (RC) (He et al., 2016) in detail and discuss how they can address the gradient problem. We first incorporate batch normalization (denoted as VGG38+BN), residual connections (denoted as VGG38+RC), and their combination (denoted as VGG38+BN+RC) to the given VGG38 architecture. We train the resulting three configurations, and VGG08 and VGG38 models on CIFAR100 (pronounced as 'see far 100') dataset and present the results. The results show that though separate use of BN and RC does mitigate the vanishing/exploding gradient problem, therefore enabling effective training of the VGG38 model, the best results are obtained by combining both BN and RC.

[1]VGG stands for the Visual Geometry Group in the University of Oxford.

2. Identifying training problems of a deep CNN

[Question Figure 3 - Replace this image with a figure depicting the average gradient across layers, for the

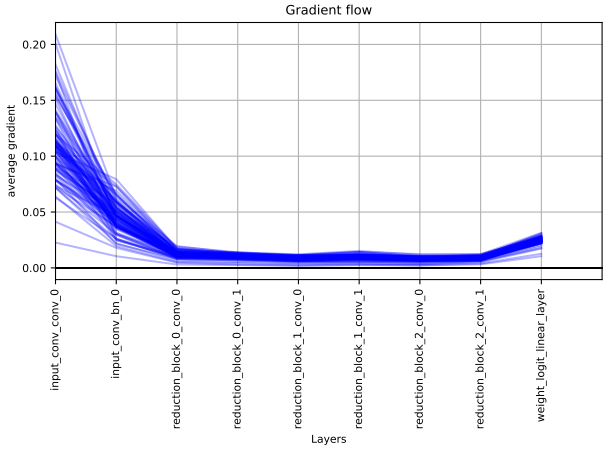


Figure 2. Gradient flow on VGG08

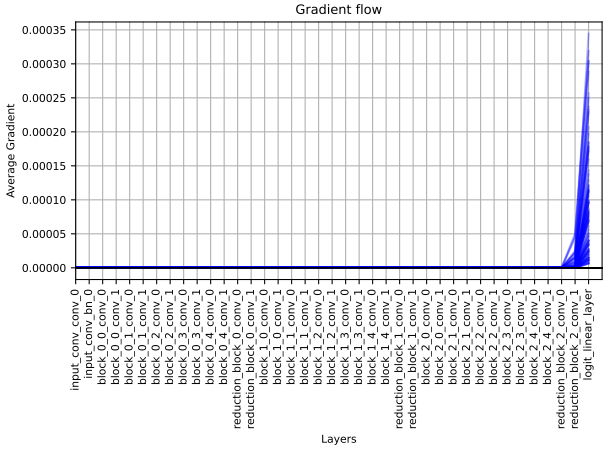


Figure 3. Gradient Flow on VGG38

VGG38 model.

(The provided figure is correct, and can be used in your analysis. It is partially obscured so you can get credit for producing your own copy).]

Concretely, training deep neural networks typically involves three steps: forward pass, backward pass (or backpropagation algorithm (Rumelhart et al., 1986)) and weight update. The first step involves passing the input $\boldsymbol{x}^{(0)}$ to the network and producing the network prediction and also the error value. In detail, each layer takes in the output of the previous layer and applies a non-linear transformation:

$$\boldsymbol{x}^{(l)} = f^{(l)}(\boldsymbol{x}^{(l-1)}; W^{(l)}) \tag{1}$$

where (l) denotes the l-th layer in L layer deep network, $f^{(l)}(\cdot, W^{(l)})$ is a non-linear transformation for layer l, and $W^{(l)}$ are the weights of layer l. For instance, $f^{(l)}$ is typically a convolution operation followed by an activation function in convolutional neural networks. The second step involves the backpropagation algorithm, where we calculate the gradient of an error function E (*e.g.* cross-entropy) for each layer's weight as follows:

$$\frac{\partial E}{\partial W^{(l)}} = \frac{\partial E}{\partial \boldsymbol{x}^{(L)}} \frac{\partial \boldsymbol{x}^{(L)}}{\partial \boldsymbol{x}^{(L-1)}} \cdots \frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{x}^{(l)}} \frac{\partial \boldsymbol{x}^{(l)}}{\partial W^{(l)}}. \tag{2}$$

This step includes consecutive tensor multiplications between multiple partial derivative terms. The final step involves updating model weights by using the computed $\frac{\partial E}{\partial W^{(l)}}$ with an update rule. The exact update rule depends on the optimizer.

A notorious problem for training deep neural networks is the vanishing/exploding gradient problem (Bengio et al., 1993) that typically occurs in the backpropagation step when some of partial gradient terms in Eq. 2 includes values larger or smaller than 1. In this case, due to the multiple consecutive multiplications, the gradients *w.r.t.* weights can get exponentially very small (close to 0) or very large (close to infinity) and prevent effective learning of network weights.

Figures 2 and 3 depict the gradient flows through VGG architectures (Simonyan & Zisserman, 2014) with 8 and 38 layers respectively, trained and evaluated for a total of 100 epochs on the CIFAR100 dataset. **[Figure 1 showed training and validation plots of shallow CNN model VGG08 vs deep CNN model VGG38 in terms of error and accuracy. Compared to the shallow model, the deep model VGG38 in Figure 1 indicated that the model was not trained at all as the loss did not decrease and the accuracy did not increase. The vanishing gradients can explain this phenomenon. Figure 3 describes well how the gradients of the loss functions for the weights in earlier layers vanish as they are propagated backwards through the network. The gradients of VGG38 grow significantly in the final layers(i.e., *logit_linear_layer*), where they are closer to the output and loss function. However, the gradients of the remaining blocks after the second reduction block are extremely small, as shown by the flat blue lines near zero. Due to the vanishing gradients, the VGG38 weights are not updated meaningfully in the earlier layers. As a result, the VGG38 model fails to learn important features in those layers, and the loss does not decrease in Figure 1. On the other hand, the shallow model in Figure 2 did not show a vanishing gradient]**.

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.

Batch Normalization (Ioffe & Szegedy, 2015) BN seeks to solve the problem of internal covariate shift (ICS), when distribution of each layer's inputs changes during training, as the parameters of the previous layers change. 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 con-

tinuously adapt to these high variance distributions which hinders the rate of convergence gradient-based optimizers. This optimization problem is exacerbated further with network depth due to the updating of parameters at layer l being dependent on the previous $l-1$ layers.

It is hence beneficial 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 BN 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 reason for BN's effectiveness is still not completely understood and it is an open research question (Santurkar et al., 2018b).

Residual networks (ResNet) (He et al., 2016) A well-known way of mitigating the VGP is proposed by He *et al.* in (He et al., 2016). In their paper, the authors depict the error curves of a 20 layer and a 56 layer network to motivate their method. Both training and testing error of the 56 layer network are significantly higher than of the shallower one.

[In general, a higher capacity by adding a layer enables the network to have more complex functions and fit more complicated data patterns. However, excessive capacity can cause overfitting as the model memorises the training data rather than generalising well on the unseen data. Interestingly, (He et al., 2016) showed a degradation of the deeper 56-layer model in both training and test performance is not due to overfitting. Instead, such degradation can be caused by optimisation difficulties commonly related to vanishing gradient problems. Deeper networks are harder to optimise due to the VGP. As the depth increases, gradients diminish as they are propagated backwards through the network, hindering earlier layers from learning effectively. The deeper 56-layer network of (He et al., 2016) experiences higher training errors as the VGP prevents the network from effective optimisations. Moreover, poor weight initialisation may have led to this disparity in training and test performance. Improper initialisation of weights in a very deep model can lead to the VGP, further hindering convergence. Additionally, simply adding more layers does not guarantee better performance. However, with architectural innovations such as residual connections, deeper models may converge to better solutions. (He et al., 2016) suggested the innovative architecture, Residual networks].

Residual networks, colloquially known as ResNets, aim to alleviate VGP through the incorporation of skip connections that bypass the linear transformations into the network architecture. 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. Prior to their work, training even a 100-layer was accepted 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.

4. Solution overview

4.1. Batch normalization

BN has been a standard component in the state-of-the-art convolutional neural networks (He et al., 2016; Huang et al., 2017). Concretely, BN is a layer transformation that is performed to whiten the activations originating from each layer. As computing full dataset statistics at each training iteration would be computationally expensive, BN computes batch statistics to approximate them. Given a minibatch of B training samples and their feature maps $X = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^B)$ at an arbitrary layer where $X \in \mathbb{R}^{B\times H\times W\times C}$, H, W are the height, width of the feature map and C is the number of channels, the batch normalization first computes the following statistics:

$$\mu_c = \frac{1}{BWH}\sum_{n=1}^{B}\sum_{i,j=1}^{H,W} \boldsymbol{x}^n_{cij} \tag{3}$$

$$\sigma_c^2 = \frac{1}{BWH}\sum_{n=1}^{B}\sum_{i,j=1}^{H,W} (\boldsymbol{x}^n_{cij} - \mu_c)^2 \tag{4}$$

where c, i, j denote the index values for y, x and channel coordinates of feature maps, and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the mean and variance of the batch.

BN applies the following operation on each feature map in batch B for every c, i, j:

$$\mathrm{BN}(\boldsymbol{x}_{cij}) = \frac{\boldsymbol{x}_{cij} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} * \gamma_c + \beta_c \tag{5}$$

where $\gamma \in \mathbb{R}^C$ and $\beta \in \mathbb{R}^C$ are learnable parameters and ϵ is a small constant introduced to ensure numerical stability.

At inference time, using batch statistics is a poor choice as it introduces noise in the evaluation and might not even be well defined. Therefore, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are replaced by running averages of the mean and variance computed during training, which is a better approximation of the full dataset statistics.

Recent work has shown that BatchNorm has a more fundamental benefit of smoothing the optimization landscape during training (Santurkar et al., 2018b) 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).

4.2. Residual connections

Residual connections are another approach used in the state-of-the-art Residual Networks (He et al., 2016) to tackle the vanishing gradient problem. Introduced by He et. al. (He et al., 2016), a residual block consists of a convolution (or group of convolutions) layer, "short-circuited" with an identity mapping. More precisely, given a mapping $F^{(b)}$ that denotes the transformation of the block b (multiple consecutive layers), $F^{(b)}$ is applied to its input feature map $\boldsymbol{x}^{(b-1)}$ as $\boldsymbol{x}^{(b)} = \boldsymbol{x}^{(b-1)} + F(\boldsymbol{x}^{(b-1)})$.

Intuitively, stacking residual blocks creates an architecture where inputs of each blocks are given two paths : passing through the convolution or skipping to the next layer. A residual network can therefore be seen as an ensemble model averaging every sub-network created by choosing one of the two paths. The skip connections allow gradients to flow easily into early layers, since

$$\frac{\partial \boldsymbol{x}^{(b)}}{\partial \boldsymbol{x}^{(b-1)}} = \mathbb{1} + \frac{\partial F(\boldsymbol{x}^{(b-1)})}{\partial \boldsymbol{x}^{(b-1)}} \tag{6}$$

where $\boldsymbol{x}^{(b-1)} \in \mathbb{R}^{C\times H\times W}$ and $\mathbb{1}$ is a $\mathbb{R}^{C\times H\times W}$-dimensional tensor with entries 1 where C, H and W denote the number of feature maps, its height and width respectively. Importantly, $\mathbb{1}$ prevents the zero gradient flow.

5. Experiment Setup

[Question Figure 4 - Replace this image with a figure depicting the training curves for the model with the best performance *across experiments you have available (you don't need to run the experiments for the models we already give you results for)*. Edit the caption so that it clearly identifies the model and what is depicted.

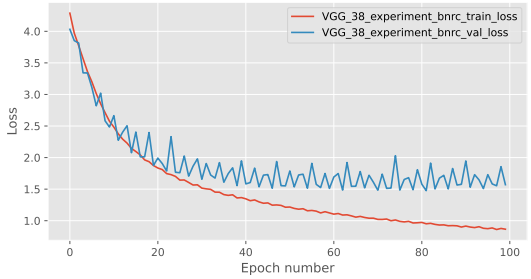
]

[Question Figure 5 - Replace this image with a figure depicting the average gradient across layers, for the model with the best performance *across experiments you have available (you don't need to run the experiments for the models we already give you results for)*. Edit the caption so that it clearly identifies the model and what is depicted.]

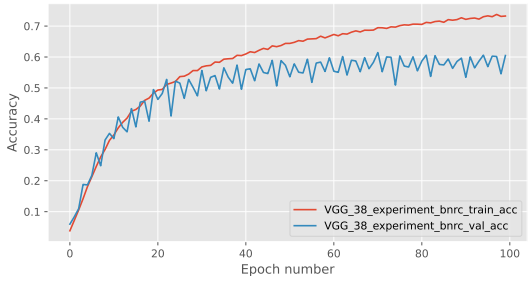
[Question Table 1 - Fill in Table 1 with the results from your experiments on

1. ***VGG38 BN (LR 1e-3)*, and**
2. ***VGG38 BN + RC (LR 1e-2).***

]



(a) Cross entropy error per epoch



(b) Classification accuracy per epoch

Figure 4. Training curves for VGG-Batch Normalisation with Residual Connections in terms of (a) cross-entropy error and (b) classification accuracy

We conduct our experiment on the CIFAR100 dataset (Krizhevsky et al., 2009), which consists of 60,000 32x32 colour images from 100 different classes. The number of samples per class is balanced, and the samples are split into training, validation, and test set while maintaining balanced class proportions. In total, there are 47,500; 2,500; and 10,000 instances in the training, validation, and test set, respectively. Moreover, we apply data augmentation strategies (cropping, horizontal flipping) to improve the generalization of the model.

With the goal of understanding whether BN or skip connections help fighting vanishing gradients, we first test these methods independently, before combining them in an attempt to fully exploit the depth of the VGG38 model.

All experiments are conducted using the Adam optimizer with the default learning rate (1e-3) – unless otherwise specified, cosine annealing and a batch size of 100 for 100 epochs. Additionally, training images are augmented with random cropping and horizontal flipping. Note that we do not use data augmentation at test time. These hyperparameters along with the augmentation strategy are used to produce the results shown in Fig. 1.

When used, BN is applied after each convolutional layer, before the Leaky ReLU non-linearity. Similarly, the skip connections are applied from before the convolution layer to before the final activation function of the block as per Fig. 2 of (He et al., 2016). Note that adding residual connections between the feature maps before and after downsampling requires special treatment, as there is a dimension mismatch

Model	LR	# Params	Train loss	Train acc	Val loss	Val acc
VGG08	1e-3	60 K	1.74	51.59	1.95	46.84
VGG38	1e-3	336 K	4.61	00.01	4.61	00.01
VGG38 BN	1e-3	338.5 K	1.74	50.95	1.91	47.35
VGG38 RC	1e-3	336 K	1.33	61.52	1.84	52.32
VGG38 BN + RC	1e-3	339 K	1.26	62.99	1.73	53.76
VGG38 BN	1e-2	339 K	1.70	52.28	1.99	46.72
VGG38 BN + RC	1e-2	338.5 K	0.86	73.24	1.57	60.48

Table 1. Experiment results (number of model parameters, Training and Validation loss and accuracy) for different combinations of VGG08, VGG38, Batch Normalisation (BN), and Residual Connections (RC), LR is learning rate.

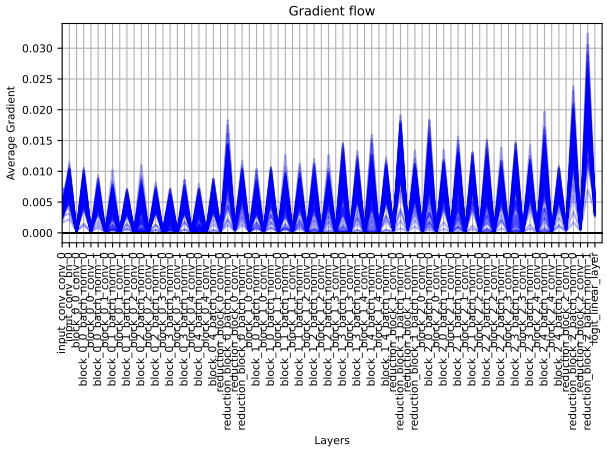


Figure 5. Gradient Flow on VGG-Batch Normalisation with Residual Connections

between them. Therefore in the coursework, we do not use residual connections in the down-sampling blocks. However, please note that batch normalization should still be implemented for these blocks.

5.1. Residual Connections to Downsampling Layers

[The downsampling process usually reduces spatial dimensionality(e.g., via pooling in the coursework), thus the dimension of the input and the output should match. To add residual connections to the downsampling layers, we achieve through two ways; projection shortcut and zero-padding (He et al., 2016).

Projection Shortcut: We can add a 1x1 convolution to the shortcut path to match dimensions. Pros can be that it ensures compatibility when dimensions differ. Also, it can improve model capacity by allowing for learnable transformations in the shortcut path. Finally, it can prevent information from being lost during downsampling. However, the Cons can be that additional convolution operations increase computational cost. Also, adding more parameters to the shortcut path can increase the risk of overfitting.

Zero-Padding: We can pad the input with zeros to match dimensions after downsampling. Pros can be that it can be computationally efficient because it doesn't operate additionally at the residual path. Also, it is easy to implement and effective for smaller dimension alterations. However, the Cons are that it can cause a mismatch if the downsampling process decreases the number of channels significantly. Also, it may not capture complicated transformations which can be required for a certain task.].

6. Results and Discussion

[In this paper, we have conducted various experiments on the CIFAR100 dataset. The best performance shown is the deeper VGG 38 network with the combination of the batch normalisation and the residual connections optimised with a 1e-2 learning rate. It showed the highest accuracy and the lowest loss on validation data, respectively 60.48% and 0.86. To achieve those returns, we controlled four factors in various experiments: model capacity, batch normalisation, residual connections, and learning rate.

As a network has a deeper architecture, it generally performs better as the increased model capacity can capture complex features and find the pattern well. However, the VGG38 model suffers from the VGP and potentially higher sensitivity to hyperparameters such as the learning rate and initialisation. The VGP can be one of the main reasons for the optimisation difficulty of the network, causing the poorest performance in a deeper network. To combat the VGP, we suggested BN and residual connections(RC), showing significant performance improvement compared to the VGG38 network, respectively 47.35% and 52.32% for the VGG38 BN and the VGG 38 RC. Combining BN and RC with the same learning rate(i.e., 1e-3) performs better at 53.76%.

Figure 4 shows that the VGG38 BN+RC model converges faster and achieves higher training and validation accuracy compared to all other models. It converges to around 1.5 validation loss at around 40 epochs although it shows a bit of irregularity, but still, the losses didn't over 2.0. Instead, the VGG38 model converges to 2.0 validation loss around 70 epochs. We could see that combining RN and RC improves convergence speed in deeper models, achieving higher performance.

Furthermore, in Table 1, the combination models show

different performances based on the learning rate. The higher learning rate represents the better performance at 60.48 %. (**Ioffe & Szegedy, 2015**) discovered that BN enables a higher learning rate by normalising activations in each layer. However, this paper comparing the learning rate 1e-3 and 1e-2 for the VGG38 BN at Table **1**, shows a different result to our expectation. We assume that deeper networks like VGG38 benefit from smaller, precise updates to weights. On the other hand, the combination model performs as we expected with the effects of RC. RC adds a direct shortcut path, allowing gradients to flow more easily through deep layers. With RC, the gradients are better stabilised and can handle the larger updates caused by a higher learning rate. Also, a larger learning rate can reduce training time by accelerating convergence in the early stages. In Figure **5**, the combination model shows the stability of gradient propagation due to the effects of BN and RC.

In this paper, we have learned that BN and RC can combat the VGP in a deeper layer. As a further experiment, we can train VGG08 with BN and RC to assess whether RC and BN still improve performance in shallow networks where the VGP is less of an issue. We expect that the impacts of RC and BN are marginal because a shallow model has fewer layers and less gradient instability. Moreover, we can test different learning rates(e.g., 1e-25, 1e-15, 1e-1, or 1e-05) for the combination model as a learning rate has a significant impact on model performance to improve the performance of the model (**Goodfellow et al., 2016**). We expected 1e-25 or 1e-15 would show the best performance in this setting because too high a learning rate can cause an overshoot of the optimal point in the loss surface while training can become unstable.] .

7. Conclusion

[In general, as a model capacity such as the depth of a model increases, optimal capacity would exist where a generalisation error is globally small as a deeper model can capture more complex features and patterns during training (**Goodfellow et al., 2016**). However, in this paper, just stacking layers like the VGG 38 network suffers from a vanishing gradient issue, showing a significantly low performance. To tackle this issue, a batch normalisation (**Ioffe & Szegedy, 2015**) and residual connections (**He et al., 2016**) are suggested in this study. Each suggestion shows satisfactory performance separately, but the combination of BN and RC performed the best performance with a 1e-2 learning rate as BN and RC work synergistically to address key challenges such as the VGP. BN stabilise training whereas RC enhances gradient flow. (**Santurkar et al., 2018a**) represents BN contributes to the surface of the loss function less steep and less irregular, instead of reducing a so-called internal covariate shift, which refers to the change in the distribution of inputs to a layer in a deep neural network during training. They only focus on the impact of BN on training, not including testing for generalisation. The effect of BN on the testing set can be our future work. Additionally, (**Santurkar et al., 2018a**) showed that BN smooths the optimisation landscape, but the VGG 38 BN does not smooth the loss function at **Appendix A** whereas the combination model(i.e., VGG38 BN + RC) smooths the optimisation landscape. We hope to investigate how RC interacts with BN to impact the geometry of the loss surface in the future.] .

References

Bengio, Yoshua, Frasconi, Paolo, and Simard, Patrice. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pp. 1183–1188. IEEE, 1993.

Bishop, Christopher M et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

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.

Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens, and Weinberger, Kilian Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

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.

Krizhevsky, Alex, Hinton, Geoffrey, et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Yann A, Bottou, Léon, Orr, Genevieve B, and Müller, Klaus-Robert. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

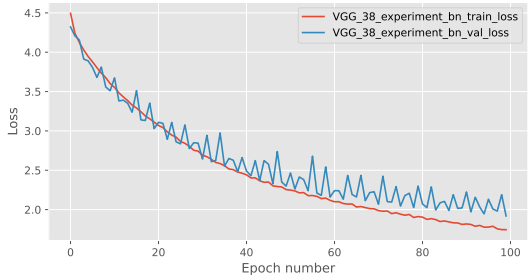
Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, and Madry, Aleksander. How does batch normalization help optimization? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2488–2498. Curran Associates, Inc., 2018a.

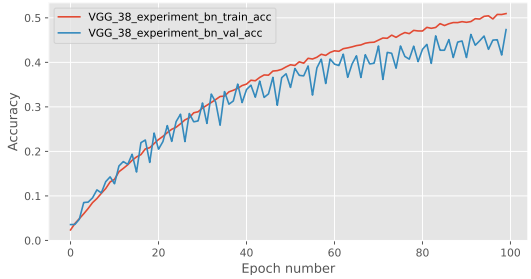
Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, and Mądry, Aleksander. How does batch normalization help optimization? In *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488–2498, 2018b.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

A. Appendix



(a) Cross entropy error per epoch



(b) Classification accuracy per epoch

Figure 6. Training curves for VGG-Batch Normalisation in terms of (a) cross-entropy error and (b) classification accuracy