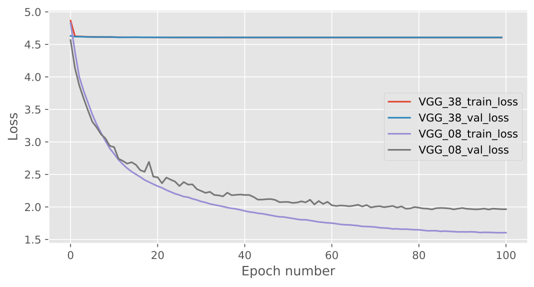
MLP Coursework 2

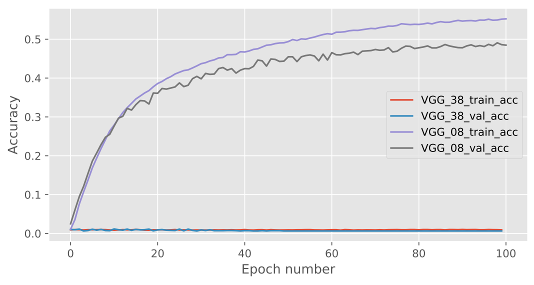
s2447408

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 (**??**), 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 (**?**), RNNs (**?**), and CNNs (**?**). Multiple solutions have been proposed to mitigate this problem by using weight initialization strategies (**?**), activation functions (**?**), input normalization (**?**), batch normalization (**?**), and shortcut connections (**??**).

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) (**?**) and residual connections (RC) (**?**) 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

Concretely, training deep neural networks typically involves three steps: forward pass, backward pass (or backpropagation algorithm (**?**)) 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

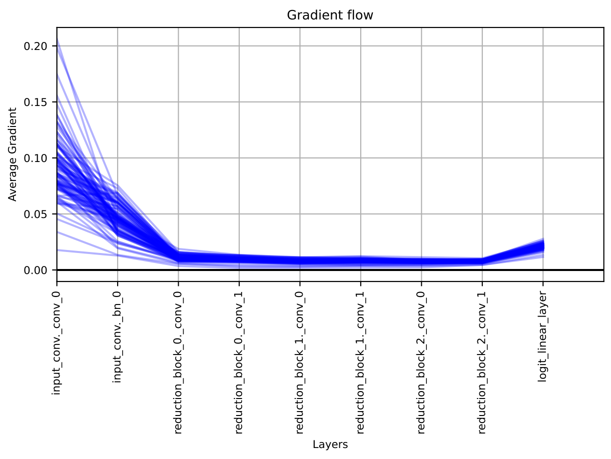


Figure 2. Gradient flow on VGG08

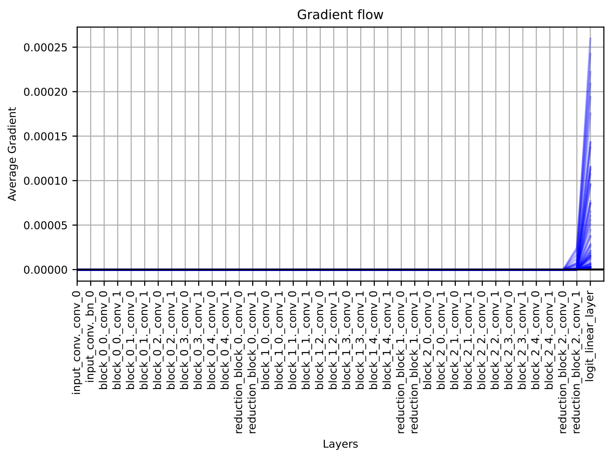


Figure 3. Gradient Flow on VGG38

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:

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

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 (**?**) 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 (**?**) with 8 and 38 layers respectively, trained and evaluated for a total of 100 epochs on the CIFAR100 dataset. **[The Vanishing Gradient Problem is present in VGG38 but not in VGG08. Figure 1 shows that both training and validation errors for VGG38 do not decrease while VGG08 shows a good decrease as the training progresses. The accuracy curve also shows a similar behaviour. Figures 2 and 3 depict the average gradients at each layer in the model. For VGG08, the gradient values are larger in the initial layers and decrease with more layers. Also, all layers have a gradient value greater than zero. But for VGG38, we see that most layers have a zero gradient except some of the final layers. This shows that when a gradient becomes zero, it is backpropagated so that all layers before it also end up with a zero gradient. From these figures we can infer that when the gradients are zero, the weights do not change and the error function does not decrease. The gradient descent never reaches a global minimum. Essentially there is no training occurring in the VGG38 model.]** .

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 (**?**) 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 continuously 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 (**?**). 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 (**?**).

Residual networks (ResNet) (**?**) A well-known way of mitigating the VGP is proposed by He *et al.* in (**?**). 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.

[A deeper network will have a higher network capacity. It should potentially have the capability to learn more complex functions. But high network capacity can also lead to overfitting as the network becomes more tuned to the training data leading to low training errors but high testing errors. But in Figure 1 from ((?)) and the results obtained above, the training error itself is very high for the deeper network. So this cannot be due to overfitting. The authors of ((?)) put this issue forward as a degradation problem, where when the depth of the network increases, accuracy saturates and then degrades rapidly. Vanishing gradients can contribute to this problem. Another factor is that deeper networks are more difficult to optimize. More layers and complex computations mean more parameters that need to be optimized. This can lead to the network getting stuck to local minima and never reaching a global minimum.].

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 (**??**). 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:

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

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

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:

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

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 (**?**) 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 (**?**).

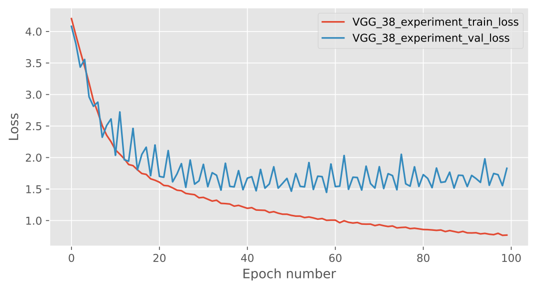
4.2. Residual connections

Residual connections are another approach used in the state-of-the-art Residual Networks (**?**) to tackle the vanishing gradient problem. Introduced by He et. al. (**?**), 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 $\boldsymbol{x}^{(b)} = \boldsymbol{x}^{(b-1)} + F(\boldsymbol{x}^{(b-1)})$.

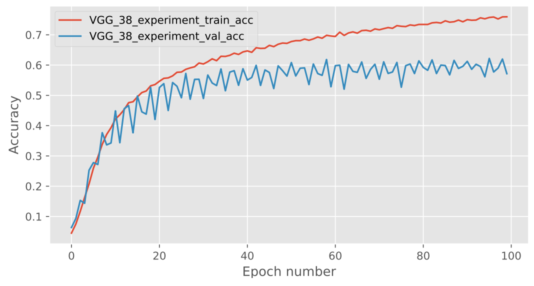
Intuitively, stacking residual blocks creates an architecture where inputs of each blocks are given two paths : passing through the convolution or skipping to the next layer. A residual network can therefore be seen as an ensemble model averaging every sub-network created by choosing one of the two paths.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



(a) Cross entropy error per epoch



(b) Classification accuracy per epoch

Figure 4. Training curves for VGG38 with Batch Normalisation (BN) and Residual Connections (RC) in terms of (a) cross-entropy error and (b) classification accuracy

feature maps, its height and width respectively. Importantly, $\mathbb{1}$ prevents the zero gradient flow.

5. Experiment Setup

We conduct our experiment on the CIFAR100 dataset (**?**), 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

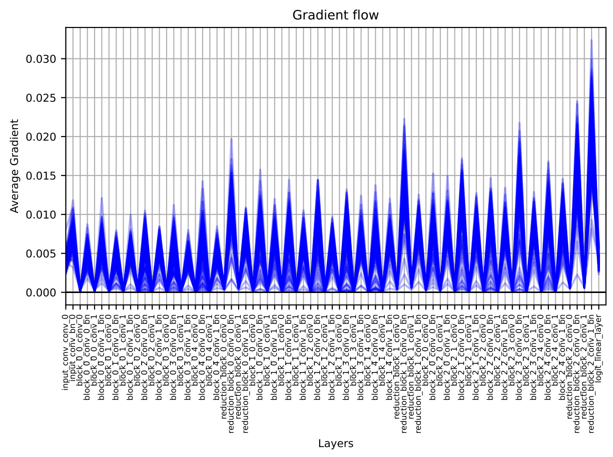


Figure 5. Gradient Flow on VGG38 with Batch Normalisation (BN) and Residual Connections (RC)

to before the final activation function of the block as per Fig. 2 of (**?**). Note that adding residual connections between the feature maps before and after downsampling requires special treatment, as there is a dimension mismatch 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

[Adding residual connections to a downsampling layer is not straightforward. Since the input and output have different shapes we cannot directly add them and form a residual connection. One way to achieve this is to apply the same downsampling operation on the identity layer(i.e., the input to block) before adding it to the output to form the residual connection. This method is easy to implement but requires more computations as we perform the downsampling twice. Also if the shapes of the output and identity do not match, we would need to use a convolution layer to correct it.
Another way would be to add a 1x1 convolution at the beginning and end of the downsampling block (before the ReLU activation). The residual connection can be added after this layer. This is called a bottleneck architecture. This ensures that the shapes of the input and output always match. Adding the 1x1 convolution reduces the number of parameters going into the convolution layers thereby reducing computation and making the model more efficient. But this method involves adding more convolutional layers to the model.] .

6. Results and Discussion

[From the experiment results in Table 1, we can infer that adding Batch Normalisation or Residual connections to the VGG38 model fixed the vanishing gradient problem. The VGG38 BN model produces non-zero gradients leading to lowering errors and provides a final

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	339 K	1.40	59.16	1.98	49.88
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	339 K	0.80	75.22	1.86	56.84

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.

accuracy of about 50% for validation data. VGG38 RC performs better than VGG38 BN with a final validation accuracy of 52%. It should also be noted that the BN model requires more parameters to be trained than the RC model. This is because batch normalisation introduces more computations in the batch normalisation layer while residual connections do not involve any extra computation layer, it reuses the computations in the previous layers. So the RC model can complete training faster. The combination of both techniques is used in VGG38 BN + RC which performs better than them individually with a validation accuracy of 53%. But the improvement is not very significant.

The gradient flow for the VGG38 BN + RC model is shown in Figure 5. We can see positive gradients on all layers. The spikes are caused by the batch normalisation layers which have a higher gradient with respect to convolution layers. But from the figure, we can conclude that the gradients are being propagated back to the initial layers uniformly in all the iterations in training.

The last two models in Table 1 use a larger learning rate of 1e-2 in training. For the VGG38 BN model, we can see that the performance has decreased i.e., the validation accuracy has come down from 50% to 47%. The degradation is more evident in the training data where the accuracy has come down from 59% to 52%. For the VGG38 BN + RC model with learning rate 1e-2, we see the best validation accuracy rate of 56%. There is also a significant improvement from the VGG38 BN model with the same learning rate. But interestingly, looking at the training curves for the VGG38 BN + RC model in Figure 4, we can see that the validation loss and accuracy fluctuate and do not remain steady, unlike the training loss and accuracy. It does achieve a better validation accuracy of 60% in the second last epoch. This could be because of the high value of the learning rate, which causes the model to bounce around the optimal solution and not converge stably. In the VGG38 BN +RC model with a learning rate of 1e-3, we can see that it has not reached the optimal solution yet, which means that the learning rate is too small.

Comparing the different models it seems that the VGG38 BN + RC gives a better performance. However, the learning rates used are not optimal. Further experiments can be performed with different learning rates. Since 1e-3 was too small and 1e-2 was too large, we can run the model with learning rates 5e-3, 2.5e-3 and 7.5e-3 to find the optimal value. Another experiment can be done to run a VGG38 RC model with learning rate 1e-2 to check if the behaviour observed with learning rate 1e-3 can be replicated where RC performs better than BN. This experiment can also validate whether the combination of BN + RC is better than only RC. If the improvement is not very significant, it can indicate residual connections alone can be used without the additional overhead of using batch normalisation.

] .

7. Conclusion

[The objective of this report was to understand the vanishing gradient problem observed in deep neural networks and identify techniques to mitigate it. We analysed the VGG08 and VGG38 models and discovered that the vanishing gradient problem occurs only in the VGG38 model. Two techniques were identified to solve this problem, Batch Normalisation and Residual Connections. We applied both these approaches in the VGG38 model and gathered the results.

The results confirmed that both batch normalisation and residual connections can help mitigate the vanishing gradient problem. The model with only residual connections performed better than the one with only batch normalisation. Batch normalisation requires more parameters in the model leading to more computations and training time. But using a combination of both techniques gives the best-performing model. We ran the models with two different learning rates, 1e-2 and 1e-3. We found that 1e-2 was too large leading to the model not converging. We also found that 1e-3 was too small since the model did not reach the optimal solution in 100 epochs.

The VGG38 model can be run with different learning rates to improve its performance. In the above experiments, we have not added residual connections to the downsampling layers. Adding this can improve the model. From Figure 1, it is also clear that there is a divergence between the training and validation accuracies/losses. This could be due to overfitting. A regularisation technique like dropout can be used to reduce the

generalisation gap.] .