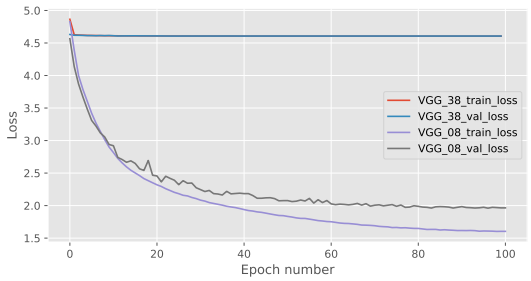
MLP Coursework 2

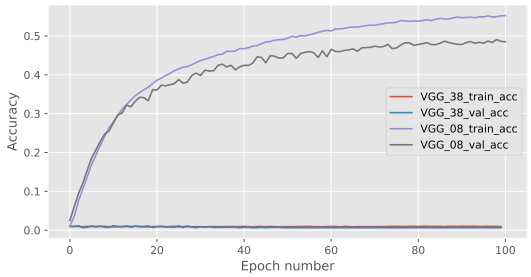
sXXXXXXX

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) Loss per epoch



(b) Accuracy per epoch

Figure 1. Training curves for VGG08 and VGG38

1. Introduction

Despite the remarkable progress of deep neural networks 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 and addressing it by implementing two standard solutions. In particular, we first study a "broken" network in terms of its gradient flow, norm of gradients with respect to its weights for each layer and contrast it to ones in the healthy 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 CIFAR-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.

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 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

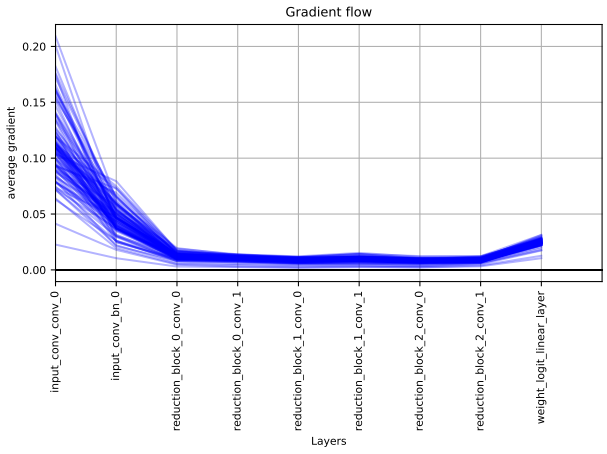


Figure 2. Gradient flow on VGG08

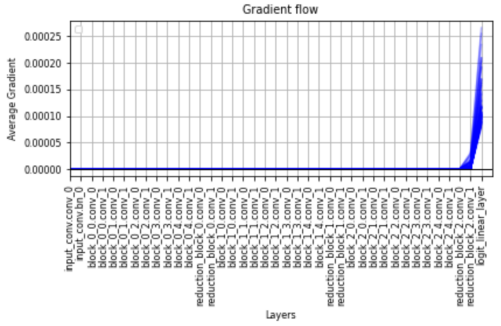


Figure 3. Gradient Flow on VGG38

$W^{(l)}$ 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 (**?**) that typically occurs in the backpropagation step when some of partial gradient terms in Eq. **??** 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 **??** and **??** 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 gradient flows are calculated by training each network for one hundred iterations. During each training iteration, the absolute mean gradients that are propagated to the upstream layer are identified and are plotted in figures 2 and three. The VGG08 network has an much larger average gradient than the VGG38 network with COMPARE DIFFERENCES.**

The gradient relationship is also inverted between the two networks where the largest gradients are propagated at the initial layers in the VGG08, whereas after 3 back-propagations in the VCG38 no informatoin is backpropogated. This prevents the weights, biases and other learnable parameters in the VGG38 network from updating and thus the network is unable to adapt the parameters to fit the data. The results are demonstrated in the training curves in figure one. There is no measurable change in the accuracy of the network or optimisation of the categorical cross entropy over the training iterations as the weights do not adapt to the data. This can be compared to the VGG08 network where both accuracy and losses change asymptotically with respect to training iterations, suggesting that that given the model architectures the weights can be updated to find the function minima (?).] .

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 layerâĂŹs inputs changes during training, as the parameters of the previous layers change. The authors argue that without batch normalization, the distribution of each layerâĂŹs 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 BNâĂŹs 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.

[Question 2 - Consider these results (including Figure 1 from (?)). Discuss the relation between network capacity and overfitting, and whether, and how, this is reflected on these results. What other factors may have lead to this difference in performance?

The average length for an answer to this question is approximately 1/5 of the columns in a 2-column page] .

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

[Question 3 - Describe Batch Normalization (BN) by using equations. Consider both training and test time and explain how BN addresses the vanishing gradient problem. Note that you are not required to provide the derivation of gradients w.r.t. weights for BN weights.

The average length of an answer to this question would be around 2/3 of a column in a 2-column page] .

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

Figure 4. Training curves for ? ? ?

Figure 5. Gradient Flow on ? ? ?

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

$$\frac{\partial \boldsymbol{x}_{(b)}}{\partial \boldsymbol{x}^{(b-1)}} = \mathbb{1} + \frac{\partial F(\boldsymbol{x}^{(b-1)})}{\partial \boldsymbol{x}^{(b-1)}} \tag{3}$$

where $\boldsymbol{x}^{(b-1)} \in \mathbb{R}^{H\times W\times C}$ and $\mathbb{1}$ is a $\mathbb{R}^{H\times W\times C}$-dimensional tensor with entries 1. 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).***

]

We conduct our experiment on the CIFAR-100 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

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	?	?	?	?	?
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	?	?	?	?	?

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.

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 Figure **??**.

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 Figure 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

[Question 4 - In this coursework, we didn't incorporate residual connections to the downsampling layers. Explain and justify what would need to be changed in order to add residual connections to the downsampling layers. Give and explain 2 ways of incorporating these changes and discuss pros and cons of each.] .

6. Results and Discussion

[Question 5 - Present and discuss the experiment results (all of the results and not just the ones you had to fill in) in Table 1 and Figures 4 and 5 (you may use any of the other Figures if you think they are relevant to your analysis). You will have to determine what data are relevant to the discussion, and what information can be extracted from it. Also, discuss what further experiments you would have ran on any combination of VGG08, VGG38, BN, RC in order to

- **Improve performance of the model trained (explain why you expect your suggested experiments will help with this).**
- **Learn more about the behaviour of BN and RC (explain what you are trying to learn and how).**

The average length for an answer to this question is approximately 1 of the columns in a 2-column page] .

7. Conclusion

[Question 6 - Briefly draw your conclusions based on the results from the previous sections (what are the take-away messages?) and conclude your report with a recommendation for future work.

Good recommendations for future work also draw on the broader literature (the papers already referenced are good starting points). Great recommendations for future work are not just incremental (an example of an incremental suggestion would be: "we could also train with different learning rates") but instead also identify meaningful questions or, in other words, questions with answers that might be somewhat more generally applicable.

For example, (?) end with

> **"Because of their compact internal representations and reduced feature redundancy, DenseNets may be good feature extractors for various computer vision tasks that build on convolutional features, e.g., [4,5]."**

while (?) state in their conclusions that

> **"There remains theoretical questions to be considered, such as whether the problem with simple gradient descent discussed in this paper would be observed with chaotic attractors that are not hyperbolic.**

The length of this question description is indicative of the average length of a conclusion section] .