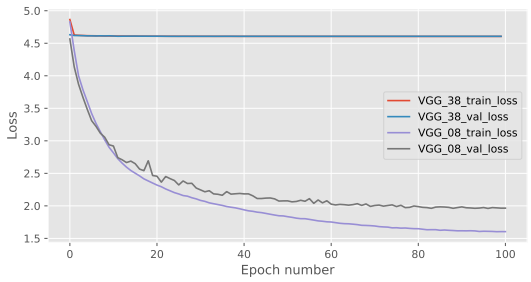
MLP Coursework 2

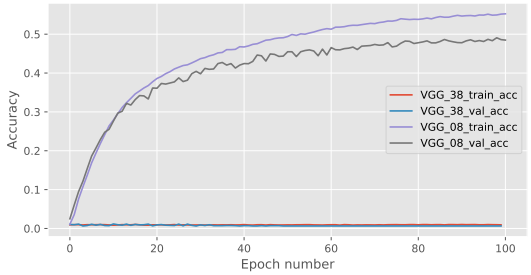
s1869292

Abstract

Deep neural networks have become the state-of-the-art in many standard computer vision problems thanks to more powerful neural networks and large labeled datasets. While very deep networks allow for better deciphering of the complex patterns in the data, training these models successfully is a challenging task due to problematic gradient flow through the layers, known as vanishing/exploding gradient problem (VGP and EGP respectively). 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 (Simonyan & Zisserman, 2014; He et al., 2016), training very deep networks is a challenging procedure. One of the major problems is the VGP, a phenomenon where gradients from the loss function shrink to zero as they backpropagate to earlier layers, hence preventing the network from updating its weights effectively. This phenomenon is prevalent and has been extensively studied in various deep network including feed-forward 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 occurred in the VGG38 model and addressing it by implementing two standard solutions. In particular, we first study the "broken" network in terms of its gradient flow, norm of gradients with respect to model 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) (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 CIFAR-100 dataset and present the results. The results show that though separate use of BN and RC does tackle the vanishing/exploding gradient problem, therefore enabling the 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 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 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

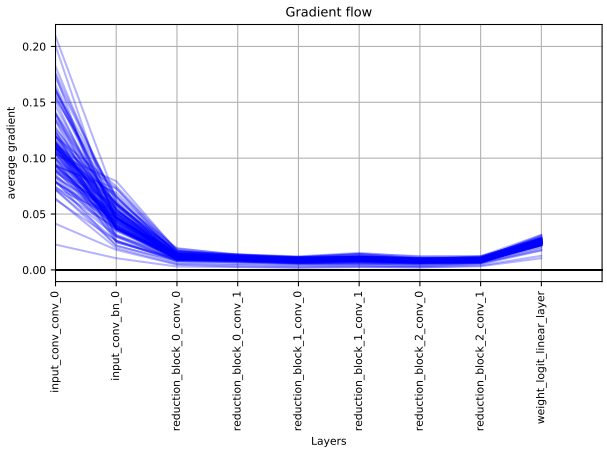


Figure 2. Gradient flow on VGG08

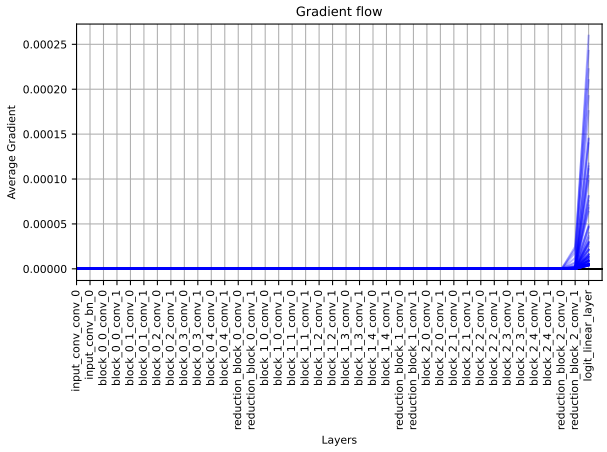


Figure 3. Gradient Flow on VGG38

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:

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

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 2 shows a "healthy" gradient flow for VGG08, meanwhile in Figure 3 we see a prototypical case of the Vanishing Gradient Problem for VGG38. Looking from the output layer (right), leftwards, we see the gradient quickly 'vanishing'; completely undetectable around layer 36 (three from output). The consequnece is shown in the Figure 1 training curves, where, VGG08 trains successfully, whereas VGG38 does not even manage to train poorly; it has completely stopped the network from training, ergo no improvement is made from its random initialisation. This totalising obstruction of the Vanishing Gradient Problem historically contributed to why early Deep Neural Networks were not feasible.] .

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 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 (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., 2018).

Residual networks (ResNet) (He et al., 2016) One interpretation of how the VGP arises is that stacking non-linear layers between the input and output of networks makes the connection between these variables increasingly complex. 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. He *et al.* observed this on a deep 56-layer neural network counter-intuitively achieving a higher training error than a shallower 20- layer network despite higher theoretical power. Residual networks, colloquially known as ResNets, aim to alleviate this 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

[As discussed in Section 3, the Batch Normalisation method (1) normalises[1] the inputs to a layer for each mini-batch, then (2) uses two new parameters, β and γ, to respectively shift (the mean) and scale (the varience); these are learnt during training, and (3) updates the backprop algorithm to account for the transformed inputs.

Concretely for a layer x of a network, where $x_i^{(k)}$ represents the k^{th} dimension of x for the i^{th} training example, we have

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \epsilon}} \tag{3}$$

where $\mu_B^{(k)}$ and $\sigma_B^{(k)^2}$ are respectively the k^{th} dimension's mean and variance (for batch B) and ϵ is an arbitrarily small constant added to preventing a division by zero.

The normalisation is then followed with the rescaling,

$$y_i^{(k)} = \gamma^{(k)} \hat{x}_i^{(k)} + \beta^{(k)} \tag{4}$$

where $\gamma^{(k)}$, the new mean and $\beta^{(k)}$, the new variance are learned along with the network weights and biases in the ongoing optimisation process. Generally β is initialised to 0 and γ to 1. This step allows the model to restore its full representational power.

[1] Sets $\mu = 1$ and $\sigma^2 = 1$, where μ is the mean and σ^2 is the varience.

Overall we have the batchnorm transform,

$$BN_{\gamma^{(k)},\beta^{(k)}} : x_i^{(k)} \rightarrow y_i^{(k)} \tag{5}$$

where m is the number of training examples in a mini-batch. Note that $\hat{x}_i^{(k)}$ stays internal to the specific layer.

Finally, the numerous derivations for the back propagation step can be found on page 4 of Ioffe & Szegedy (2015).

Although Batch Normalisation was not designed specifically to combat the VGP, it is a powerful normalisation and regularisation technique with many use cases. One way it addresses the VGP is by transforming the input to a range that will result a healthy set of gradients for activation functions that 'prefer' inputs within a specified range. For example, the Sigmoid function returns minuscule gradients for numbers of moderately large absolute value (say ±3) and beyond. Its worth noting that batchnorm's lone effectiveness is limited for very deep networks (especially with squishifying activation functions) as repeated multiplication (as taken in gradient backprop) of values less that |1|, will tend towards zero] .

4.2. Residual connections

[ResNets overcome the tendency of networks to degrade in performance[2] past a certain depth (classically 15-30 layers). Residual neural networks do this by utilising skip connections, or shortcuts to jump over some layers.

Formally, given a weight matrix of the form $W^{\ell-k,\ell}$ connecting layer $\ell - k$ to ℓ (in our experiments we have stuck to $k = 2$), the forwardprop step of our ResNet block is written,

$$\begin{aligned} y^{\ell} &:= g\left((W^{\ell-1,\ell} \cdot y^{\ell-1} + b^{\ell}) + y^{\ell-k}\right) \\ &= g\left(\mathcal{F}^{\ell} + y^{\ell-k}\right) \end{aligned} \tag{6}$$

where y^{ℓ} is the output of layer ℓ, g is the activation function, $W^{\ell-k,\ell}$ the weight matrix connecting layer $\ell - k$ to ℓ and $\mathcal{F}^{\ell}(= W^{\ell-1,\ell} \cdot y^{\ell-1} + b^{\ell})$ is the standard connection before introduction of the shortcut. Note $\mathcal{F}^{\ell} + y^{\ell-k}$ is performed by element-wise addition. If the dimensions of the layers $\ell - k$ and ℓ are not equal (as is common in convolution networks due to pooling), we do not add residual connections, however it would be possible to perform a linear projection onto the new dimensions as suggested in the original paper (He et al., 2016).

A major benefit of ResNets is that they add no parameters or computational complexity. Similarly, the back propagation step is the same for both standard and skipped pathways, essentially unchanged from its classical formulation,

$$\begin{aligned} \Delta w^{\ell-k,\ell} &: = -\eta \frac{\partial E^{\ell}}{\partial w^{\ell-k,\ell}} \\ &= -\eta y^{\ell-k} \cdot \delta^{\ell} \end{aligned} \tag{7}$$

[2] training performance included; ergo the model is not suffering from overfitting.

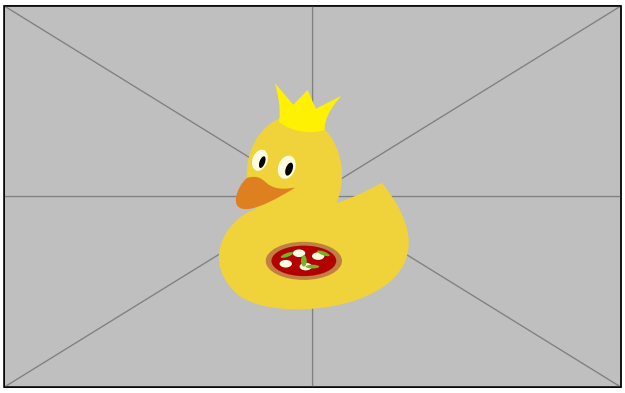


Figure 4. Training curves for ? ? ?

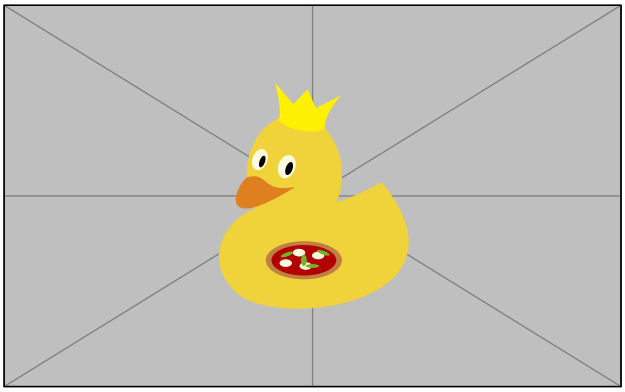


Figure 5. Gradient Flow on ? ? ?

where η is the learning rate, y^ℓ the output from layer ℓ, and δ^ℓ the error at layer ℓ. As back propagation is of the same form for both the standard and skipped path, the weight matrices are merged and learned in one step (at no additional cost)] .

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. (Also edit the caption accordingly).]

[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. (Also edit the caption accordingly).]

[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 (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 Figure 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 Figure 2 of (He et al., 2016)

6. Results and Discussion

[ ] .

7. Conclusion

[Question 5 - 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, (Huang et al., 2017) 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 (Bengio et al., 1993) 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

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	?	?	?	?
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.

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] .

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.

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

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