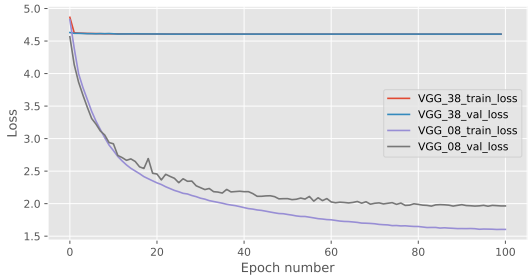
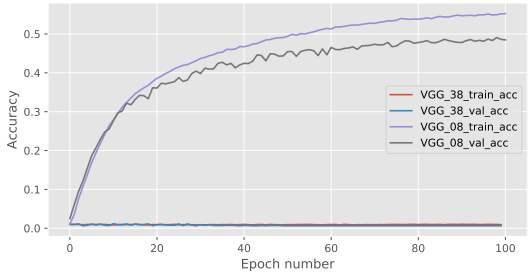
Analysis of Vanishing gradient problem

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

[Question Figure 3 - Replace this image with a figure depicting the average gradient across layers, for the VGG38 model.]

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

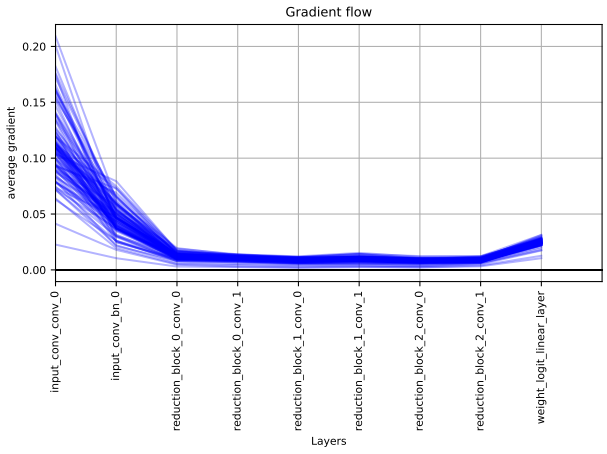


Figure 2. Gradient flow on VGG08

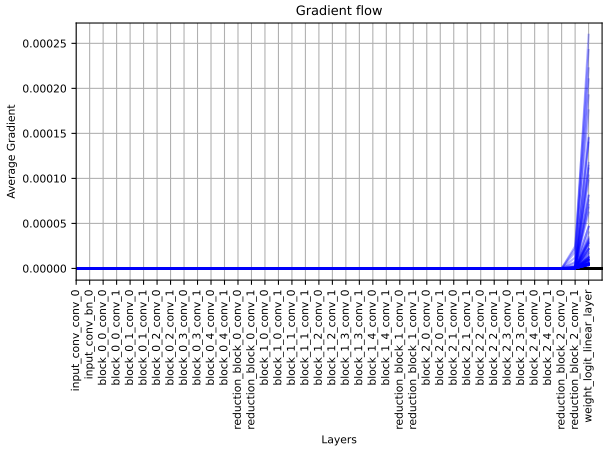


Figure 3. Gradient Flow on VGG38

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. **[We can observe that from fig 3 the model faces the above mentioned problem of vanishing gradient because at layer REDUCTION_BLOCK_2_CONV_0 we see that the mean absolute gradient value approaches zero and remains same for all the hidden layers. This leads to exponentially small or no updates in weights of hidden layers during backpropagation. This observation also supports the reason as to why the VGG38 model has no updates in loss or accuracy(in fig 1) as the number of epochs increase. This proves that the model VGG38 is broken and is not able to learn new features during training as compared to VGG08 which shows expected behaviour.]** .

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

[According to the original paper (Ioffe & Szegedy, 2015) in batch normalization during training we take each mini batch of size m and B such mini batches. According to the authors we use (β, γ) as learnable parameters. Average of sample means of B mini batches with size m -

$$E(x) = E_B(\mu_B) \tag{3}$$

Average of sample variance of B mini batches with size m -

$$var(x) = \frac{m}{m-1} \cdot E_B(\sigma_B^2) \tag{4}$$

Using γ and β as learnable parameters by the BN layer we get the following equation for batch normalization at train time.

$$y = \frac{x - E(x)}{\sqrt{var(x) + \epsilon}} \cdot \gamma + \beta \tag{5}$$

During test time we use the learned parameters (β, γ) and calculate output of BN layer as follows-

$$y = \frac{\gamma}{\sqrt{Var(x) + \epsilon}} \cdot x + \left(\beta - \frac{\gamma \cdot E(x)}{\sqrt{Var(x) + \epsilon}}\right) \tag{6}$$

As mentioned earlier we don't completely understand how BN works ((Santurkar et al., 2018)) but we can argue that by using BN we reduce the internal covariance shift by normalizing the inputs throughout the layer for each mini batch, before feeding them into activation function. We do this before sending output to activation layer because we do not want the lose any information obtained by the layer after the activation. and then passing them to the next layers so that during back prop the gradient will be less dependent on the scale of the parameters or the assigned initial values. With the help of BN we can also use high learning rates to train models faster without any ill effects.] .

4.2. Residual connections

[According to authors (He et al., 2016) residual connections for each building block of CNN is defined as follow-
'y' represents the output of building block with 'i' number of CNN layers in block

$$y = F(x, \{W_i\}) + x \tag{7}$$

The above equation when used for two CNN layers in a block will have the follow notation

$$y = W_2\sigma(W_1 * x) + x \tag{8}$$

During test time we use the same representation for skip connections.
Residual connections are used to mitigate the problem of vanishing or exploding gradient as they create a short cut connection for each block of CNN layers and help pass the gradient values easily during the back prop step. The authors of the paper (He et al., 2016) did not intend to apply RC to solve Vanishing gradient problem(VGP) but help the network learn better and faster. In results it turned out that RC also solve the VGP by allowing the gradient to pass through hidden layers with the help of identity transform and help the network learn better and faster compared to baseline case where there are no RC in the network.] .

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).***

]

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-2	339 K	1.70	52.28	1.99	46.72
VGG38 BN	**1e-3**	**339 K**	**1.82**	**49.39**	**2.12**	**43.92**
VGG38 RC	1e-3	336 K	1.33	61.52	1.84	52.32
VGG38 BN + RC	1e-3	339 K	0.626	80.43	1.661	59
VGG38 BN + RC	**1e-2**	**339 K**	**0.524**	**83.22**	**1.85**	**61.36**

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.

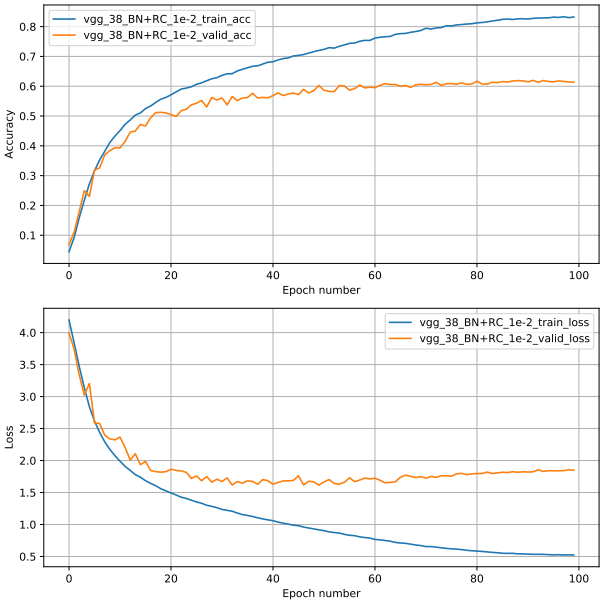


Figure 4. Training curves for VGG38 BN + RC(1e-2)

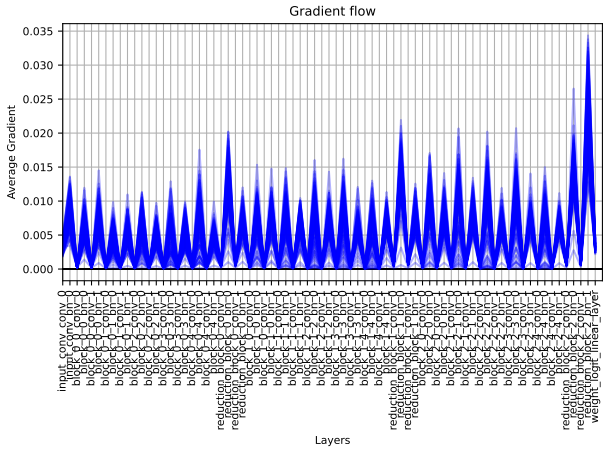


Figure 5. Gradient Flow on VGG38 BN + RC(1e-2)

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

[From table 1 we observe that after implementing Batch Normalization(BN) the vanishing gradient problem is solved and VGG _38 network starts to learn new features . When experimented with different learning rates we find that the results are good when the learning rate was increased from 1e-3 to 1e-2 for same number of epochs and this is to be expected because BN helps us to facilitate high learning rates withoug having ill affects.(NOTE for the experiments with BN we did not implement BN in reduction convolutional block but only implemented it in convolutional block)
When Residual Connections(RC) were implemented in the VGG_ 38 network we can observe that it solves the problem of vanishing gradient as well and also we get high train and validation accuracy compared to BN(table 1).(NOTE: For residual connections it is applied only convolutional block and not in reduction block).

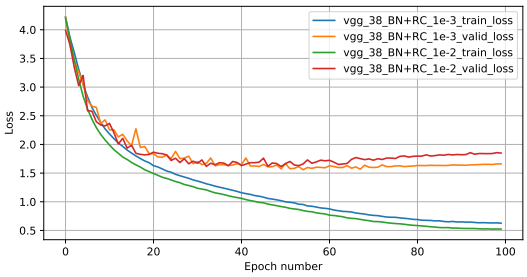


Figure 6. Loss curves for VGG _38 network with LR(1e-2,1e-3)

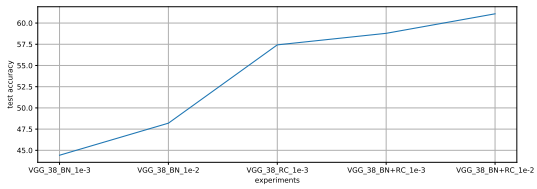


Figure 7. Comparision of test accuracies for different experiments

To further improve the accuracy of the model we try a combination of BN and RC for different learning rates and observe that the network with BN and RC with learning rate 1e-2 gives the highest test and validation accuracy. We can assert from fig 5 that the gradient value passes throughout the network easily and hence gives us good results.(NOTE in these combination experiments BN is applied to convolutional block and reduction block but RC is applied to only convolutional blocks.)
When we compare the results of different networks from table 1 on test data (fig 7) we find that the model which we selected as a best one gives us the highest test accuracy of 61.09%.(fig 7)
It is interesting to note that from fig 4 the best model we selected is overfitting to training data inspite of giving high test accuracy. So we can do further experiments to improve the test accuracy by adding regularization techniques to the network. In this paper ((Chen et al., 2019)) the authors discuss about new layer which is a combination of Batch Normalization and Dropout layers know as IC(Independent Component) layer. By using this mechanism we can apply regularization to our VGG _38 network and can expect better results because the authors in the paper has tested it in on CIFAR 10/100 datasets using ResNet's with IC and found that the results were much better than the baseline cases.
To further understand what is actually happening in BN and we can follow practices mentioned in this paper((Summers & Dinneen, 2020) where the experiments where done on ResNets and this may help us gain further insights as how to improve the performance of BN.] .

7. Conclusion

[We can conclude from our experiments that even though Batch Normalization and Residual Connections individually solve problem the problem of Vanishing gradient, it is observed that we can find better accuracy when we use a combination of them. We also found out that the best model which is a combination of BN+RC faces problem of overfitting when learning rate is increased (fig 6) and to solve these problems we proposed some existing solutions. In future, experiments we can try using the proposed solutions in the final part of sec 6 in combination we can use new activation function ELU(Exponential Linear Units) instead of RELU as proposed in this paper((Thakkar et al., 2018) because of its promising results wtih VGG model in comparision to DenseNet, ResidualNet and Inception(v3) on CIFAR 10 dataset.] .

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.

Chen, Guangyong, Chen, Pengfei, Shi, Yujun, Hsieh, Chang-Yu, Liao, Benben, and Zhang, Shengyu. Rethinking the usage of batch normalization and dropout in the training of deep neural networks, 2019.

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.

Summers, Cecilia and Dinneen, Michael J. Four things everyone should know to improve batch normalization, 2020.

Thakkar, Vignesh, Tewary, Suman, and Chakraborty, Chandan. Batch normalization in convolutional neural networks — a comparative study with cifar-10 data. In *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, pp. 1–5, 2018. doi: 10.1109/EAIT.2018.8470438.