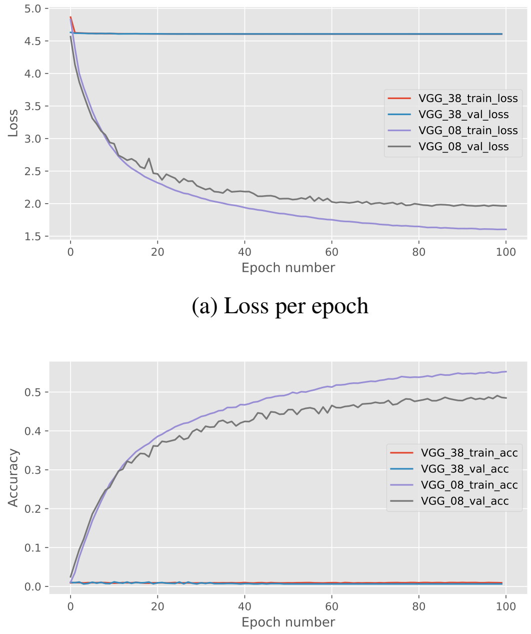
MLP Coursework 2

s2153223

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., 2016a), 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 feedforward networks (Glorot & Bengio, 2010), RNNs (Bengio et al., 1993), and CNNs (He et al., 2016a). 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., 2016a; 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., 2016a) 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

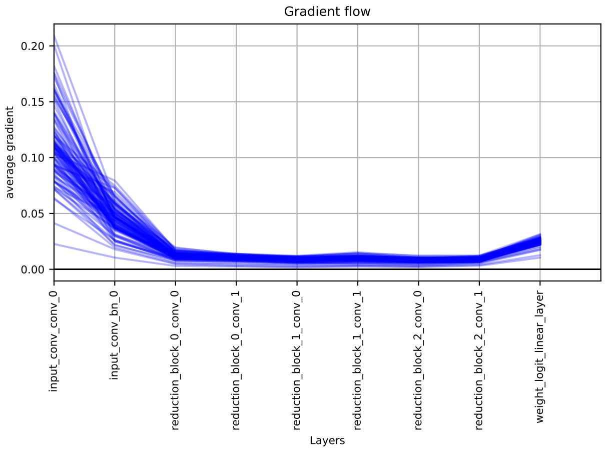


Figure 2. Gradient flow on VGG08

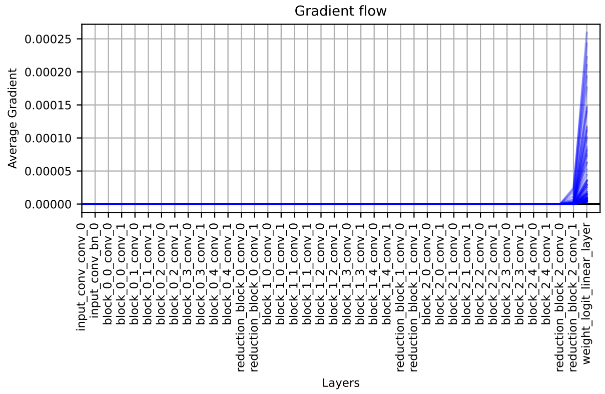


Figure 3. Gradient Flow on VGG38

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. **[In the gradient flow of VGG38 in Figure 3, the average gradients are 0 until the last reduction block and do not change. At the same time, it can be found from Figure 1 that the loss and accuracy curves of VGG38 are straight lines parallel to the axis, indicating that the parameters are no longer updated and the gradient is 0.By contrast, VGG08 does not have the above problem. Therefore,we can identify that there is a Vanishing Gradient Problem in VGG38.]** .

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., 2016a) 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

[Batch Normalization addresses the problem of internal covariate shift by regulating the output of each layer to the same mean and variance, so that the distribution of the input value of each layer of the neural network is pulled back to the standard with a mean value of 0 and a variance of 1. The input in Batch normalization is Values of x over a mini-batch:$\mathcal{B} = \{x_{1...m}\}$, parameters to be learned are γ, β. And the process of Batch normalization is as follows, mainly including normalizing and scaling and shifting:

$$mini-batchmean : \mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3}$$

$$mini-batchvariance : \sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \tag{4}$$

$$normalize : \hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{5}$$

$$scale\ and\ shift : y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \tag{6}$$

The normalized result is obtained by calculating the mean and variance of the mini-batch, and then scale and shift are performed.Finally we obtain output$\{y_i = BN_{\gamma,\beta}(x_i)\}$.

In the process, batch normalization pulls the output down from the saturated zone to the unsaturated zone, thereby addressing the vanishing gradient problem. In addition, due to batch normalization uses the sample mean and variance computed over the complete training data as the mean and variance parameters for each layer, higher learning rates are allowed and the training time of the neural network is reduced.] .

4.2. Residual connections

[Residual Connections address the degradation problem by using shortcut connections to let layers fit a residual mapping $\mathcal{F} := \mathcal{H}(x) - x$ instead of fitting $\mathcal{H}(x)$.The original function thus becomes $\mathcal{F}(x) + x$, and it would be easier to push the residual to zero.

Residual Connections can also address the vanishing gradient problem.When adopting residual learning to every few stacked layers,a building block is defined as:

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

Here x and y are the input and output vectors of the layers considered. $\mathcal{F}(x, \{W_i\})$ represents the residual mapping to be learned, W_s is a linear projection.

In the process of backpropagation, we can calculate the gradient of the loss function:

$$\frac{\partial loss}{\partial \boldsymbol{x}_l} = \frac{\partial loss}{\partial \boldsymbol{x}_L}\frac{\partial \boldsymbol{x}_L}{\partial \boldsymbol{x}_l} = \frac{\partial loss}{\partial \boldsymbol{x}_L}\left(1 + \frac{\partial}{\partial \boldsymbol{x}_l}\sum_{i=1}^{L-1}\mathcal{F}(\boldsymbol{x}, \{W_i\})\right) \tag{8}$$

It can be found from the equation that $\frac{\partial loss}{\partial x_l}$ is not less than 1. The vanishing gradient problem can thus be addressed. And the time complexity of each layer is roughly the same. Residual Connections has almost no influence on training time and test time.] .

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

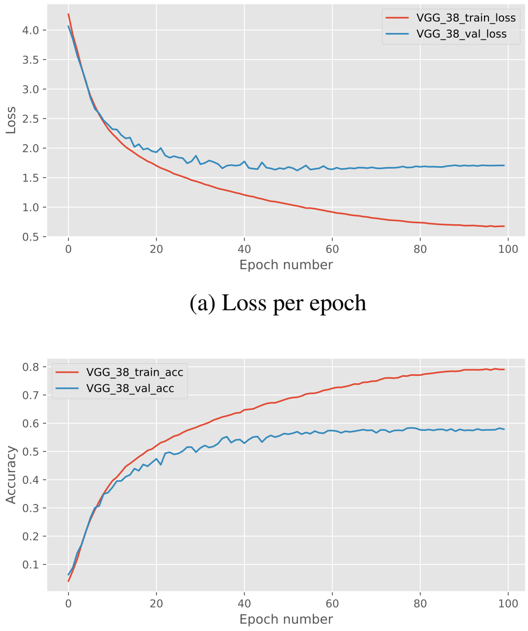
1. ***VGG38 BN (LR 1e-3)*, and**
2. ***VGG38 BN + RC (LR 1e-2)*.**

]

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

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	1.56	55.66	1.95	46.56
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	0.79	67.70	1.71	57.88

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.



(a) Loss per epoch

(b) Accuracy per epoch

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

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.

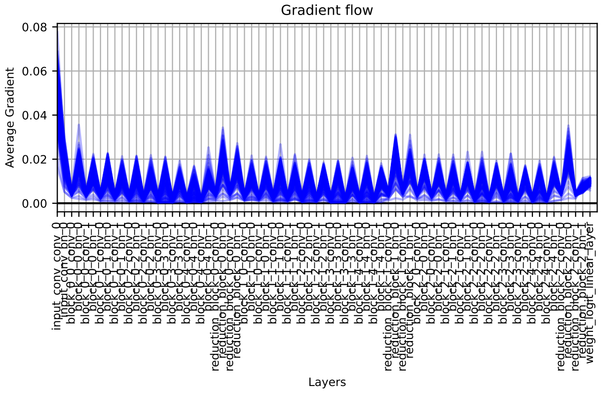


Figure 5. Gradient Flow on VGG38(BN+RC),lr=1e-2

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., 2016a)

6. Results and Discussion

[From Table 1,we can find that VGG38(BN+RC), lr=1e-2 has the best performance with the training accuracy of 67.70 and the validation accuracy of 57.88. The training curves in Figure 4 and the gradient flow in Figure 5 show that the vanishing gradient problem has been addressed. The average gradient flows between 0.00 and 0.04, and the parameters are constantly updated, so that the accuracy is continuously improved and the loss is continuously reduced.

In addition, after using BN and RC, the training accuracy and validation accuracy are both above 46. Compared with the VGG38 with the training and validation accuracy of 0.01, indicating that the problem of gradient disappearance has been improved.

The performance of VGG38 (BN+RC),lr=1e-3 is the second, and the accuracy is close to that of lr=1e-2. VGG38 (RC) followed closely, and the validation accuracy reached 52.32. VGG38(BN) was slightly worse, but the validation accuracy also reached 46 or more.

Experiment results have proved that both BN and RC can address the vanishing gradient problem, and the model with lr=1e-2 has a higher accuracy than the model with lr=1e-3. The effect of RC is slightly better than that of BN, and the average gradient flow is more gentle. When BN and RC are used at the same time, better results can be obtained.

In the above experiment, BN is applied after each convolutional layer, before the Leaky ReLU non-linearity, and the skip connections are applied from before the convolution layer to before the final activation function of the block: convolutional layer+ BN + Leaky ReLU + addition. After referring to the experiments by (He et al., 2016b) in section 4, we think the order of RN, RC and Leaky ReLU may affect the model. In the next step, we will re-arrange the activation functions (Leaky ReLU and/or BN) to improve performance of the model trained.

First, change the location where skip connections are added. Set the addition in front of the second Leaky RelU or the second BN. Next, without changing the position of skip connections, change the order of convolutional layer, BN and Leaky ReLU: Leaky ReLU + convolutional layer + BN + addition and BN + Leaky ReLU + convolutional layer + addition.

By experimenting with the four new models, we can explore the impact of the order of BN and RC and explore whether changing the order of them can improving the model. At the same time, I can learn more about the usage of activation functions, including the difference between BN after addition, Leaky ReLU before addition, post-activation and pre-activation in RC.] .

7. Conclusion

[We first analyze the vanishing gradient problem in VGG models with 8 and 38 hidden layers on the CIFAR100 image dataset by monitoring the gradient flow during training, and identify a vanishing gradient problem in the VGG38 model with the validation accuracy of 0.01. Then, we use batch normalization and residual connections to address the problem and do experiments to compare the effects of the two solutions. Experiments show that both solutions can effectively solve the problem, and the training accuracy and validation accuracy are all above 46. The effect of residual connections is slightly better than that of batch normalization. When the two are combined, the effect is the best with the validation accuracy of 57.88. In addition, the model with learning rate of 1e-2 has a higher accuracy than the learning rate of 1e-3. This enable a deeper model to outperform shallower ones in the same experimental setup.Further experiments may focus on re-arranging the activation functions (Leaky ReLU and/or BN) to improve performance of the model trained. We plan to study the combination of some variants of residual connections and batch normalization to make the training time of the model shorter, and to improve the model.] .

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, 2016a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks, 2016b.

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.