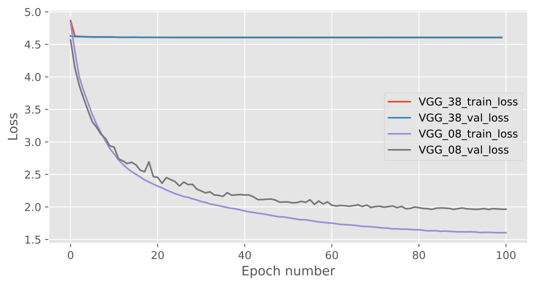
MLP Coursework 2

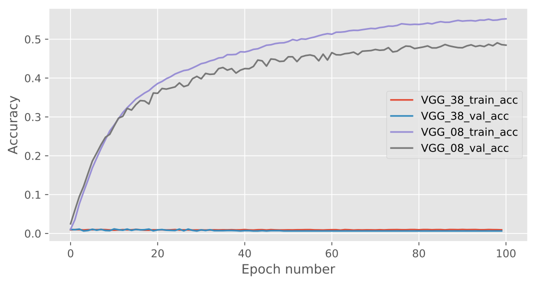
s1862323

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 (Simonyan & Zisserman, 2014; He et al., 2016), 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 (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 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) (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 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

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

***(The Figure we give is correct, and can be used in your analysis. It is partially obscured so you can get credit for producing your own copy).*]**

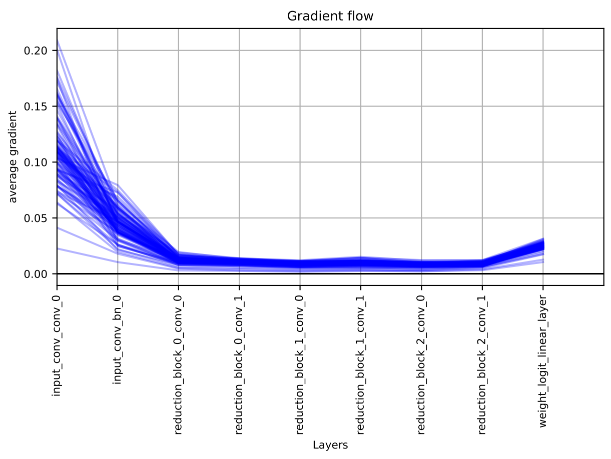


Figure 2. Gradient flow on VGG08

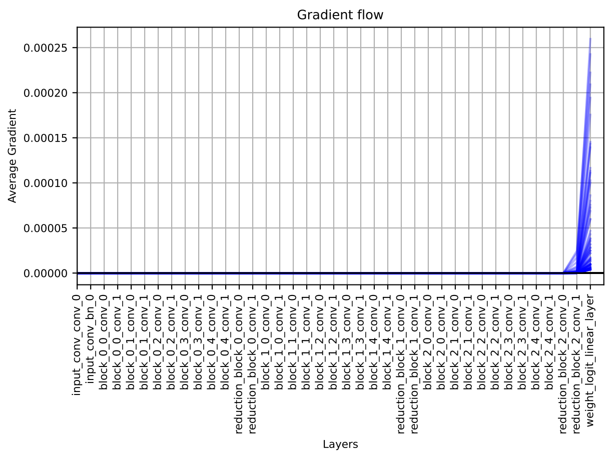


Figure 3. Gradient Flow on VGG38

Concretely, training deep neural networks 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 $\boldsymbol{x}^0$ 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 $W^{(l)}$ 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. **[The VGG38 model suffers from a vanishing gradient problem and the accuracy of this model didn't improve as shown in the Figure. Figure 1 shows that the VGG38 train loss and valid loss maintain around 4.5 until the end. In contrast, the VGG08 train loss and valid loss decrease as the epochs. The accuracy shows a similar situation, VGG08 train accuracy and valid accuracy increase as the epochs but the VGG38 didn't improve. By comparing Figure 2 and Figure 3 I found VGG08 has a higher average gradient than VGG38 in the first convolution layer. VGG38 has almost 0 average gradient, which means VGG38 model is facing VGP problem and hamper convergence. Overall, the VGG38 model has a VGP problem. It results in the model that is not trained well, and the training accuracy and validation accuracy didn't improve.]** .

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) A well-known way of mitigating the VGP is proposed by He *et al.* in (He et al., 2016). 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.

[Figure 1 from (He et al., 2016) shows deeper network has higher training error and test error. Deeper networks expose degradation problem which is not caused by overfitting. Because adding more layers to a suitably deep model leads to higher training error. The construction solution can make the added layers as identity mapping and ensure no higher training error. However, the author addresses the degradation problem by introducing a deep residual learning framework. ResNet method can easily enjoy accuracy gains from greatly increased depth, and is easy to optimize. Overall, ResNet method solves the degradation problem, and makes the deeper layer has higher accuracy and lower training error.] .

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

[Batch Normalization based on mini-batch and do normalization for different activation functions. Batch Normalization is regularizer, in some cases eliminating the need for Dropout. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. The equation of BN is shown below.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

```
$\mu_\beta \leftarrow \frac{1}{m}\sum_i^m x_i$                                  // mini-batch mean
$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_i^m (x_i - \mu_\beta)^2$              // mini-batch variance
$\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$  // normalize
$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$          // scale and shift
```

By adding Batch Normalization before the activation function, BN uses the mean and standard deviation of small batches to continuously adjust the intermediate output of the neural network, making the intermediate output values of the entire neural network layers more stable. BN want to ensures that |x| lies within the good range and avoid the gradient vanish before the activation function.

In addition, BN can speeds up convergence. With the BN, the learning rate can be adjusted to be larger, and training with a larger learning rate increases the speed of training without the upper gradient exploding when the learning rate is too large, and the lower gradient disappearing when the gradient is smaller. Once the inputs to each layer have become similarly distributed, a larger learning rate can be used (but generally without changing the accuracy of the model).] .

4.2. Residual connections

Residual connections are another approach used in the state-of-the-art Residual Networks (He et al., 2016) to tackle the vanishing gradient problem. Introduced by He et. al. (He et al., 2016), 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

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

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.

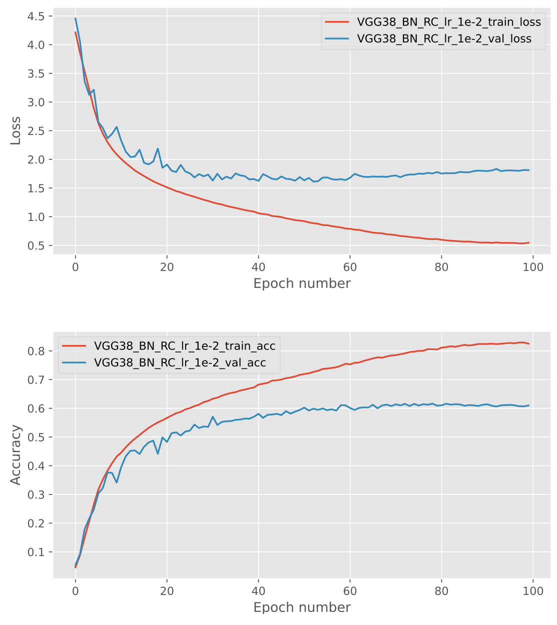


Figure 4. Training curves for VGG38-BN-RC(learning rate = 1e-2)

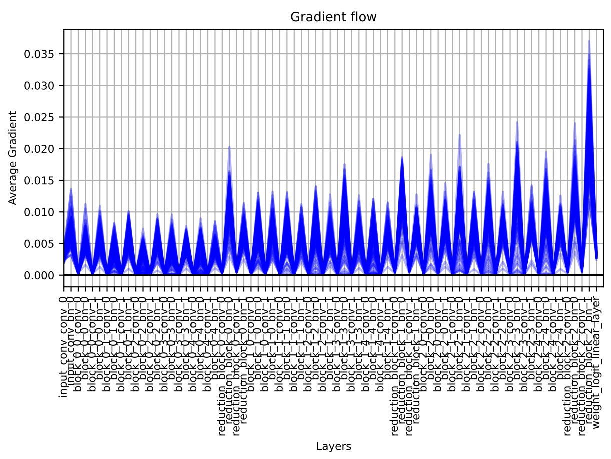


Figure 5. Gradient Flow on VGG38-BN-RC(learninng rate=1e-2)

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 (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). 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

[There are two ways to incorporate residual connections to the downsampling layers. (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. (B) 1x1 convolutions can be used to change the dimensionality to match dimension. . Pros of method A: this option introduces no extra parameter; This option has computational benefit since it is trivial to accomplish. Cons: Adds 0 at the edges and could lead to veils. Pros of method B: the filter reduces dimensionality across channels. Cons: Increase the computational cost.

] .

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	1.62	53.81	1.99	45.52
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	0.62	80.69	1.75	60.00

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.

6. Results and Discussion

[Based on the observation from table 1, although VGG08 doesn't expose the VGP problem, the accuracy of this model is not large enough. Because the VGG08 has only 8 layers and I want to increase the depth of the layer to improve the model performance. Because VGG38 has the VGP problem and training accuracy and validation accuracy of the model are almost zero, which causes model to stagnate. In order to prevent the VGP problem, VGG38-BN model is supposed to solve the VGP problem, when the learning rate is 1e-3, the validation accuracy increases to 45.52%. And when the learning rate is 1e-2, the validation accuracy increase to 46.72%, which is a slight improvement when we adjust the value of learning rate. In addition, the VGG38-RC model is supposed to solve the degradation problem. When the learning rate of VGG38-RC model is 1e-3, the validation accuracy increase to 52.32%. In order to further improve the VGG model performance, we decided to use VGG38-BN-RC model. The VGG38-BN-RC(learning rate = 1e-3) has better performance in training accuracy and validation accuracy. Hence, the VGG38-BN-RC is a better choice than single using BN or RC combine with VGG38 model.And, when the learning rate becomes larger (1e-2), the VGG38-BN(1e-2) model has better performance than VGG-BN(1e-3).

In order to improve the performance of the model. I reference the observation mentioned above. I decide to choose a large learning rate(1e-2) and the combined model(VGG-BN-RC) to improve the VGG38 model performance. As Figure 4 shown above, the training loss and the validation loss of VGG38-BN-RC have improved, which is 0.54 and 1.81 respectively. Also, this combined model with learning rate(1e-2) has the best model performance compared to others(Table 1). Moreover, the training accuracy and validation accuracy also are the best among other models (Figure 4). Because the VGG38-BN-RC is a better choice than single using BN or RC combine with VGG38 model. And, when the learning rate becomes larger (1e-2), the VGG38-BN(1e-2) model has better performance than VGG-BN(1e-3).

As Figure 5 shows above, the average gradients are mostly greater than zero. The gradient didn't vanish as the shown in Figure 3. Because this model applied the Batch normalization and solved the VGP problem. In addition, as Figure 5 shown, the final layer gradient(weight$_{l}ogit_{l}inear_{l}ayer)increasestoabetterrangecomparedtothe$
$BNmodel.Becausethismodelappliedtheresidualconnectionsaswell.Ther$

7. Conclusion

[Batch normalization can address the vanishing gradient problem when the model has deeper layers. And the resident connections can address the degradation problem and make the model increase the accuracy as the increased depth. If we apply batch normalisation and residual connectivity to the VGG08 model, will this model also gain accuracy?] .

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.