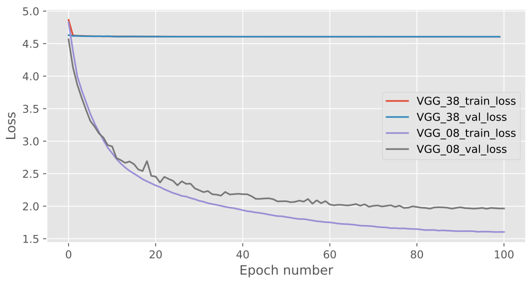
MLP Coursework 2

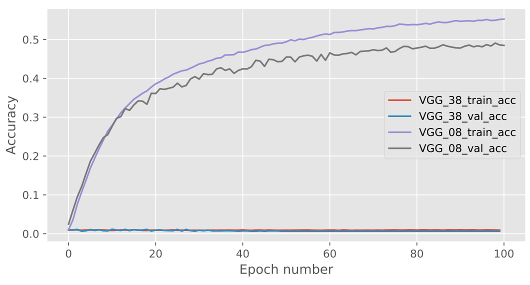
s2298839

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).*]**

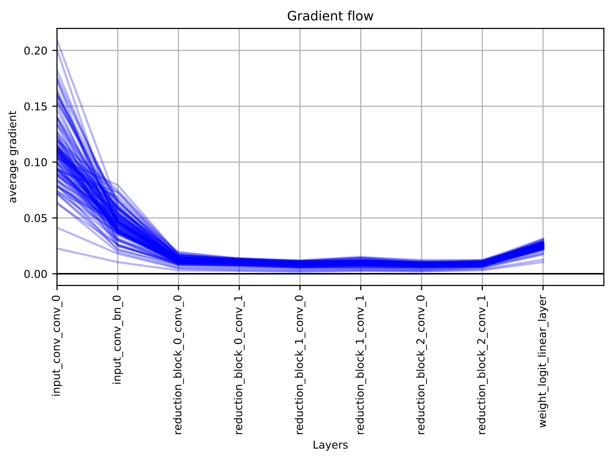


Figure 2. Gradient flow on VGG08

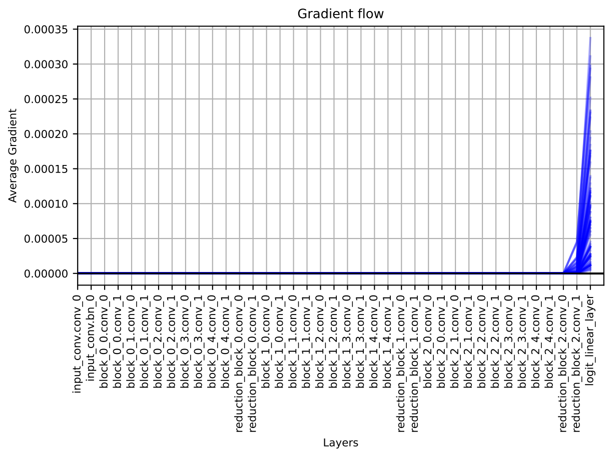


Figure 3. Gradient Flow on VGG38

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:

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

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. **[According to Figure1, as the number of epochs increases, the training and Validation loss of VGG08 decreases while training and Validation accuracy increases. However, the the training and Validation loss of VGG38 remain the same, and the training and Validation accuracy of VGG38 stay at 0, which means the model of VGG38 has no improvement in the training process.**

According to Figure2 and Figure3, the absolute mean gradient of VGG08 can be passed to the input layer. While the gradient of VGG38 only has positive values around the output layer, and no gradient is passed to the input layer through backpropogation. So the VGG38 model suffers from Vanishing Gradient Problem.

The consequences dipicted are that when simply stacking layers in network(VGG38 here) the gradient of layers(expect the last few layers) will vanish, resulting that the model can not be updated(due to 0 grad), and the model will lose the ability to learn(convergence to a low error and high accuracy relative to the initial training period).] .

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.

[According to Table1, Figure2 and Figure3 in CW1, when the network capacity(depth or width) increases, the model will suffer from overfitting, in which case the gap between training error and validation(or test) error will be bigger, and the gap between training accuracy and validation(or test) accuracy will also become bigger. This means when overfitting happens, the trained model will have a good performance on fitting the training data, but having a relatively poor ability to fit the validation(or test) data(it can be reflected on the gap between training and validation error(and acc) curves, when network capacity increases, gap bigger).
But considering the results of Figure1. The gap between training and validation error is not increasing when network capacity increases, but rather the network having deeper layers(38 to 8) will have higher training error and lower training accuracy.

Also considering the Figure1 from (He et al., 2016). Both training and validation error of 56-layer plain network are higher than 20-layer network, but the gap between training error and validation error is not increasing as network capacity increases(depth grows), and the trend performs the same in Figure4(left) from (He et al., 2016). This means overfitting does not happen.

The problem of vanishing gradients leads to the worse performance on deeper network. It happens at the beginning of convergence. When the network depth increases, accuracy gets saturated and then degrades(not caused by overfitting), which generate higher 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

[In (Ioffe & Szegedy, 2015) it describes the goal of the Batch Normalisation is to solve the problem of Internal Covariate Shift by unifying the distribution of data in each mini-batch(too costly to fully whitening the inputs of one layer). Assunming that values of x over a mini-batch with size m, $\mathcal{B} = x_{1...m}$. The output $y_i = BN_{\gamma,\beta}(x_i)$ with parameters γ, β to be learned.

The procedure of Batch Normalisation Transform is as below.
The first step is to calculate the mean and variance on each column(feature)

$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$

Then perform normalization on x_i

$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$

where ϵ is a constant added to mini-batch variance for numerical stability. If ϵ is neglected, the normalized activations will have same mean of 0 and variance of 1 at training time.
Finally y_i is computed as a linear transformation of x_i as

$y_i = \gamma x_i + \beta \equiv BN_{\gamma,\beta}(x_i)$

When considering the training and testing of Batch Normalisation networks. The x inputed to one layer is now transformed to $BN(x)$. Both batch gradient descent and SGD with mini-batch can be used to do the training.

The complete procedure of training and testing(validating, inference) a BN network is shown below.
Considering the inputs are the network N with trainable parameters Θ and subset of activations $\{x^{(k)}\}_{k=1}^{K}$, the outputs are trained BN network for testing N_{BN}^{inf}

The first step is to train the network N_{BN}^{tr} from N. So on each column(feature) k, add the transformation $y^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to N_{BN}^{tr} as the transform shown above. Then modify each layer in N_{BN}^{tr} with input $x^{(k)}$ to take $y^{(k)}$ instead. After the for-loop, Train N_{BN}^{tr} to

optimize the parameters Θ and $\{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$

After the training period, the trained network N_{BN}^{tr} will be transformed to N_{BN}^{inf} for testing by freezing its parameters.

Because the normalization of activations (depending on mini-batch data) is not needed during test time(the output should depend only on the input). So when the network has beed trained, the normalization should not contain mean and variance of one specific mini-batch, instead, population(mean and variance of the whole input) is used.

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}] + \epsilon}}$$

On each column(feature) k, to obtain the $E[x^{(k)}]$ and $Var[x^{(k)}]$, we should process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them.

$$E[x^{(k)}] = E_{\mathcal{B}}[\mu_{\mathcal{B}}^{(k)}] \qquad Var[x^{(k)}] = \frac{m}{m-1} E_{\mathcal{B}}[{\sigma_{\mathcal{B}}^{(k)}}^2]$$

Upon $\hat{x}^{(k)}$ is obtained. In N_{BN}^{inf}, replace the transform $y^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ with $y^{(k)} = \gamma^{(k)} x^{(k)} + \beta^{(k)}$

The Batch Normalisation transforms the inputs of activation to Standard Normal Distribution, which prevents the data to move to the saturated region of the non-linear activations(it will cause the gradient to vanish). For example, (Ioffe & Szegedy, 2015) mentioned the sigmoid activation will suffer from vanishing gradient if not using BN in deep network. Also, network with Batch Normalisation can have higher learning rate(because the learning rate value does not affect the backpropogation), which will to some extent prevent the problem of gradient vanishing.

] .

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

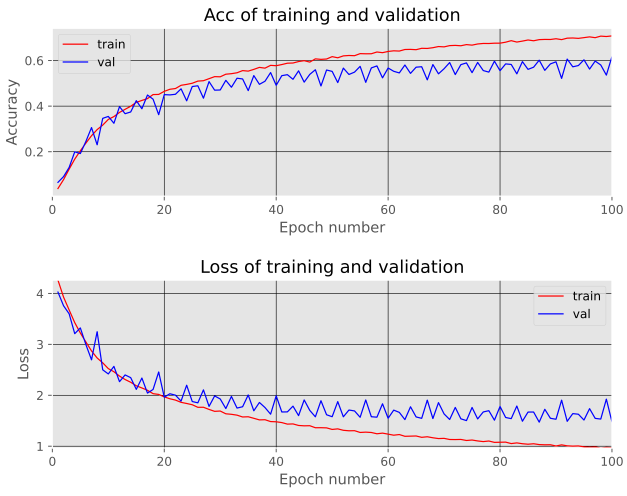


Figure 4. Training curves for BN+RC network with 1e-2 learning rate

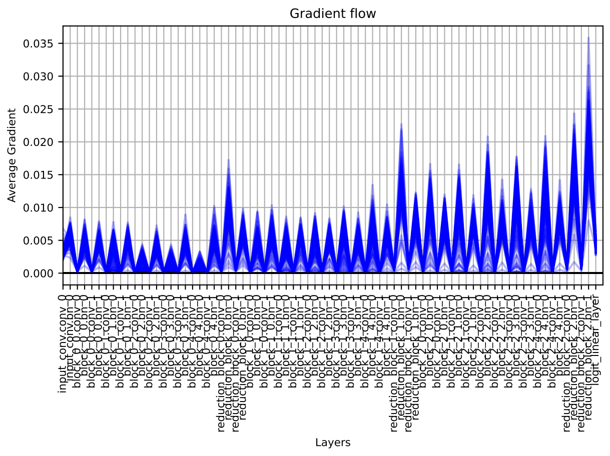


Figure 5. Gradient Flow on BN+RC network with 1e-2 learning rate

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 (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 down-sampling 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

[*Eqn.(1).* in (He et al., 2016) explains that the input x and $\mathcal{F}$ must be of same dimension so that the output y can be the sum of them. When downsampling is applied, by which the dimension of $\mathcal{F}(x, \{W_i\})$ is changed, the dimension of x must also be changed to the same as $\mathcal{F}$ in order to add residual connections to the downsampling layers.

One way to solve the dimension mismatch problem to use zero-padding shortcuts. For example the input of one block has 64 feature map dimension, while the output has 128. Then zero-padding can be performed on the input to increase dimension to 128 and match the size of the output.

Another way is to use a projection shortcuts. According to *Eqn.(2).* in (He et al., 2016), a linear projection W_j can be performed on x to fit the size of $\mathcal{F}$, so the output $y = \mathcal{F} + W_j x$

Considering the pros and cons of these 2 solutions. The strengths of zero-padding is that it is parameter-free(no additional parameters exist when padding), which can ensure a relatively small size of the model(low network capacity). This can reduce training time complexity(less parameters to calculate) and prevent overfitting(based on the results in CW1 of the relationship between network capacity and overfitt). While the shortcomings of zero-padding is that it indeed has no residual learning, instead just adding 0 to fit the output dimension, which can result in the relatively bad performance on the trained model(based on Table3 in (He et al., 2016)) compared to projection shortcuts.

The advantages of projection is that extra parameters are introduced into the network by projection shortcuts, and with the help of residual learning through backpropogation(brought by parameters for projection), the network can fit a more complex model/distribution in deep learning, which ensures a relatively lower training and validation error and higher accuracy(Table3 in (He et al., 2016)). While the disadvantages of projection is that the additional parameters need more time to train and more space to save the network, which increases the time/memory complexity. For example in bottleneck architecture described in (He et al., 2016), the parameters will be doubled if replacing identify shortcuts with projection. Also, in deep network, the parameters in projection may hamper information propagation and lead to optimization problems. Moreover, the increse of network can cause overfitting problem in some cases.

] .

6. Results and Discussion

[Comparing the experiment results in Table1. The VGG08 plain trained model has $60K$ parameters, VGG38 plain trained model has $336K$ parameters. When using BN on VGG38, the number of parameters will increase by about $3K$, while using RC does not affect it. Assunming the better performance means higher Train and Val acc, with lower Train and Val loss.

- Plain network: When comparing VGG08 to VGG38 without BN or RC, the performance of VGG08 overwhelm that of VGG38. The VGG38 suffer from problem of gradient vanishing(Figure 1,2,3) because of the zero gradient and zero trainin and validation accuracy.

- The influence of BN and RC: When only using BN for the broken VGG38, the performance of it is a little bit better than VGG08. When only using RC for broken VGG38, the performance is better than that of only using BN. This means either BN or RC can to some extent solve the problem of gradient vanishing in deep network. Though using RC

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	1.67	52.61	1.94	47.28
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	**0.97**	**70.80**	**1.48**	**61.72**

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.

can ensure better performance, yet the gap between Train(both loss and acc) and Val will be bigger than using BN alone and VGG08, which means there exists some overfitting problem in VGG38+RC network.

When combining BN and RC, the performance of VGG38 BN+RC improves a little compared to using RC alone, moreover, the overfitting problem is slightly alleviated(gap between train and val decreases). What can be inferred is that both BN and RC can handle the problem of gradient vanishing, BN is helpful is releasing the problem of overfitting, while RC ensures the network a better performance. Morever, VGG38 BN+RC can combine their strengths(low loss high acc, and the gap between train and val will not be too large).

- Change of learning rate: The performance of VGG38 BN with 1e-3 LR is slightly better than that of VGG38 BN with 1e-2 LR, which means the BN can maintain the resilience of model to the LR scale. The small decrease in Val accuracy of large learning rate is also corresponding to Figure2(BN-x5 better than BN-x30) in (Ioffe & Szegedy, 2015).

  However, the performance of VGG38 BN+RC with 1e-2 LR improves a lot compared to VGG38 BN+RC with 1e-3 LR(the best trained model so far). This may be because when LR increases, both BN and RC can make the model converge faster, so the VGG38 BN+RC with 1e-2 LR can achieve much better performance than the same model with 1e-3 LR within 100 epochs.

According to Figure4, the training curve of the best trained model VGG38 BN+RC with 1e-2 LR. It can be viewed that the validation error, though fluctuating, converges after epoch 80. And considering the Figure5, the gradient of each bn layer is not zero, while the gradient of convolutional layers are zero. So it indicates that the Batch Normalisation layers help the model to pass gradient through backpropogation, which solves the problem of vanishing gradient.

The further experiments can be planned as follows:

- Firstly, train the model of VGG08 BN, VGG08 RC, VGG08 BN+RC(the same as VGG38) with 1e-3 LR, to demonstrate what is the best trained model(just like we have selected the best trained model using VGG38 is VGG38 BN+RC).

- Secondly, after selecting the setting of the best trained model based on VGG08 and VGG38 (maybe VGG08 BN+RC and VGG38 BN+RC), choose different LR(like Figure2 in (Ioffe & Szegedy, 2015)), for example, 1e-4, 5e-4, 5e-3, 5e-2, to test what LR is best suitable for the two settings.

So in the first step, we selected the best combination of BN and RC on VGG08, then we test different learning rate to find the best LR. If in the first step, the best trained model is VGG08 plain, then we need extra experiments(do step two) on the settings which performs only worse than VGG08 plain in step one. Because since we have the prior knowledge(also can be inferred from previous experiments) that the plain network is sensitive to learning rate, so if the 1e-3 LR happens to be best suitable for plain network, then step two on only plain VGG08 is futile(the best performance maybe in VGG08 BN+RC with some other LR)

After that we shall do some experiments on learning about the behaviour of BN and RC.

- Firstly, we want to know in BN networks, whether the gradient value is negatively correlated to the LR(Since (Ioffe & Szegedy, 2015) has described that the increase of LR will cause the increase of parameter scale, and large parameters scale generates small gradients). So I will use either VGG08 BN or VGG38 BN(no RC, only study BN) with different LR(like 1e-4, 5e-4, 5e-3) and plot the gradient flow.

- Secondly, we want to know in RC networks, the different performance between zero-padding shortcuts and projection shortcuts on downsampling layers(including time/memory cost, val accuracy) and weighing the pros and cons of the two shortcuts to arrive at the most suitable combination. So I will use two VGG38 RC(no BN, only study RC), one with zero-padding on its downsampling layers,

another with projection on its downsampling layers. Train them and compare the running time, memory used and val accuracy.

] .

7. Conclusion

[We firstly find the vanishing gradient problem(VGP) in VGG38 network. Then we go over the theory of 2 typical methods BN((Ioffe & Szegedy, 2015)) and RC((He et al., 2016)), which help solve VGP. Then we do experiments on VGG38 with several combination of BN and RC. By comparing the performance and gradient flow of them, we conclude that the BN and RC is of help in solving VGP. The trained model with best performance in out experiments in VGG38 BN+RC with 1e-2 LR.

The introduction of shortcuts in RC (He et al., 2016) prevents the gradient flow in deep network not to reducing to 0, thus maintain the update of network. While the bn layers in BN network (Ioffe & Szegedy, 2015) normalize the input mini-batch, which enables layers share the same distribution, thus reducing internal covariate shift. The normalization also helps keep the layer input at the non-saturated region of the activation layers, which prevents VGP. The bn layers can also maintain the resilience of the network to parameter scale, thus large LR can be used.

Some of our experiments demonstrates same trend with others. Like the VGG38 BN with 1e-3 LR compared to VGG38 BN with 1e-2 LR. It shows similar results with Figure2 in (Ioffe & Szegedy, 2015), the BN-x5 and BN-x30.

Either BN or RC can handle VGP. BN has additional effect on solving overfitting problems(gap of using BN alone between train and val is bigger than that of using RC alone) while RC ensures a better performance of the trained model(in Table1, the performance of using RC alone is better than that of using BN alone), and both of them can increase the resilience of the network to big learning rate(LR).

Considering the future work, we could increase the number of layers in the RC network, and then test whether BN can help solve the problem of overfitting by using one network with RC+BN while the other with RC alone.

Inspired by (Ioffe & Szegedy, 2015), when the size of training data is extremely large, the batch normalization need a big batch to enable its accurate approximation of the global mean and global variance, which put pressure on hardware. So is there an alternative that can help reduce hardware requirements for training. Also, when each input instance in a batch varies from each other a lot, the BN may not help because it only computes the mean value within a batch. We can try to realize the instance normalization by only calculate the mean of one instance.

] .

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.