MLP Coursework 2: Investigating the Optimization of Convolutional Networks-Vanishing Gradient and degradation problem

s2036028

Abstract

Vanishing gradient issue and degradation problem are identified as the optimization problems that VGG38 training encountered. After reviewing related literature, we implemented the Batch Normalization mechanism at first, which effectively enabled broken VGG38 to begin converging. However, the achieved training accuracy−0.572 still worse than much shallower VGG08 and exposed a degradation problem. By integrating Residual Learning Framework further, the training accuracy rapidly increased to 0.862 and easily beat the baseline. Finally, by handling overfitting further, we achieved a stable test accuracy that 0.651 ± 0.0048.

1. Introduction

1.1. Motivation-A Counterintuitive Phenomenon

Due to the increasing requirements in visual object recognition, Convolutional Neural Networks (CNNs) have gained great attention and recent works (Szegedy et al., 2014) reveal that network depth is of crucial significance for handling complex tasks. Theoretically, as the network depth going up, CNNs are supposed to be able to gain benefits from the abstraction power, enriched features provided by extra layers, at least it should not perform worse when the shallower model nested into it (He et al., 2016). However, as shown in Figure 1, the deeper VGG38 unexpectedly performs worse than the much shallower VGG08. Surprisingly, such phenomenon is not caused by overfitting, for it is also observed on training set. Motivated by this counterintuitive phenomenon, this paper aims to explore the optimization problems behind it.

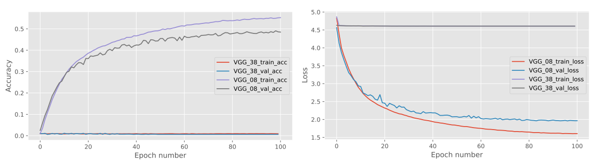
1.2. Overview

In the first place, this paper recognizes the vanishing gradient problem and degradation problem as reasons that lead to the counterintuitive phenomenon. Vanishing gradient is a well-known problem in deep networks optimization, which jeopardizes the convergence from beginning. In this paper, vanishing gradient problem was identified by visualizing the mean absolute gradient of VGG38's each layer. However, degradation problem, which refers to the unexpected lower training accuracy caused by optimization difficulty in deep network, is relatively hard to identify for it usually hides behind the vanishing gradient problem. **Therefore, in this paper, the vanishing gradient problem was solved at first. Then the degradation problem can be exposed and identified, by comparing the training performance of VGG08 and VGG38 (with repaired gradient).**

Secondly, to achieve a better understanding and find the possible solutions for the identified problems, three literatures are reviewed (Ioffe & Szegedy, 2015),(He et al., 2016), (Huang et al., 2017). Except Dense connection, both BN and RL were adopted to solve the two identified problems respectively. BN was originally proposed to solve "internal covariate shift" problem, but by normalizing the activation it can deal with vanishing and gradient problem, for it prevents the small variation of a layer's weights from being magnified too much (gradient exploding) or decreased to zero (gradient vanishing). Residual learning was developed for solving degradation problem exactly, by fitting optimization-friendly residual mapping, and it can also address vanishing gradient problem for the short connections can facilitate the flow of information through the network.

Adopting residual learning technique can solve the two problems at once, but the degradation problem hidden behind the vanishing gradient issue will never be exposed. Therefore, BN was applied at first, and then RL was introduced to address the remaining degradation problem. The experiments were done on the CIFAR100 dataset, which consists of 60,000 32×32 color images with 600 images contained in each of 100 classes. By adopting BN, the gradient of VGG38 network was repaired, but the semi-healthy VGG38 still struggling to gain accuracy from its depth and can not beat shallower model, which implies the appearance of degradation problem. Finally, a totally healthy VGG38 was established by further introducing RL technique, then the capacity of VGG38 was fully recovered and can easily beat VGG08 and solely BN-based VGG38 in terms of training accuracy, validation accuracy and convergence speed. After solving the overfitting problem,the test accuracy of the final model reported on multiple seed is 0.651 ± 0.0048.

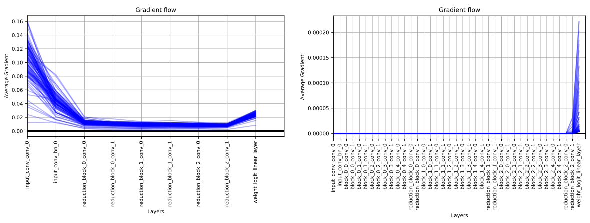
2. Identifying training problems of a deep CNN



(a) Accuracy Curves (b) Error Curves

Figure 1. Performance Comparison:VGG08 and Broken VGG38

Figure 1 exposes a counterintuitive phenomenon that VGG38 performs much worse than VGG08, when it was supposed to gain benefit from its depth and at least perform better on training set. To be more specific, optimization of VGG08 is normal, for both of its training and validation accuracy keep increasing and errors keep decreasing. However, for VGG38, neither its error nor its accuracy has improved, and both error curves and training curves are almost a straight line. It is also not lead by overfitting, for the abnormal phenomenon is observed on training set either. Back to the Figure 1, the straight line-like training accuracy curve means the VGG38 even not begin converging, which implies something going wrong during back propagation. Following this intuition, to visualize the back propagation procedures of training, the absolute mean gradient of each layers were plotted, which is the best way to quantitatively analyze the potential issues that may occur in back propagation procedure.



(a) Gradient Flow of VGG08 (b) Gradient Flow of VGG38

Figure 2. Gradient Flows of VGG08 and Broken VGG38

As shown in Figure 2, Compared with VGG08's gradient flow, where all layers have non-zero gradients and gradient flowed to the input layer successfully, the inverse-mode gradient signal or error signal of VGG38 decreases rapidly, almost disappears at the penultimate convolutional (CONV) layer and just the first few epochs has non-zero gradient at this layer, which indicates a severe vanishing gradient problem. The reasons cause the vanishing gradient problem is shown below, where g represents all kinds of non-linear activation functions, $A^{(n)} = w^{(n)h^{(n-1)}}$ and $h^{(n)} = g(A^{(n)})$ represent the n^{th} layer's activation its output respectively.

$$\begin{aligned}\Delta w^{(n)} &= \frac{\partial \text{Loss}}{\partial w^{(n)}} = \frac{\partial \text{Loss}}{\partial h^{(N)}} \frac{\partial h^{(N)}}{\partial h^{(N-1)}} \frac{\partial h^{(N-1)}}{\partial h^{(N-2)}} \cdots \frac{\partial h^{(n+1)}}{\partial h^{(n)}} \frac{\partial h^{(n)}}{\partial w^{(n)}} \\ &= \frac{\partial \text{Loss}}{\partial h^{(N)}} * g'(A^{(N)}) w^{(N)} * \cdots * g'(A^{(n+1)}) w^{(n+1)} * h^{(n-1)} \\ &= \frac{\partial \text{Loss}}{\partial h^{(N)}} \prod_{i=n+1}^{N} g'(A^{(i)}) \prod_{i=n+1}^{N} w^{(i)} * h^{(n-1)}\end{aligned} \tag{1}$$

From equation (1) it can be concluded that, depending on the activation function and the magnitude of weights, the closer a layer get to the input, the smaller or larger the gradient of its weights will be, and when network is deep enough, its gradient will be very large or approach to zero, which corresponding to the vanishing gradient and exploding gradient problem respectively. The $\prod_{n+1}^{N} g'(A^{(n)})$ item shows Relu can help mitigating vanishing and exploding gradient problem , for derivative of Relu is equal to one. However, in this paper leaky Relu is adopted to avoid "dead Relu" problem, which has derivative $\alpha = 0.01$ (default by torch) when activation is negative. Meanwhile, due to the existence of $\prod_{n+1}^{N} w^{(n)}$ item, when the network is deep, the small magnitude of weights will also lead to vanishing gradient problem, this may be the reason why Relu can't totally solve the problem.

On the basis of the analysis above two phenomena need to be explained further. Firstly, in Figure 2 (b), VGG38's gradients directly disappear at the penultimate CONV layer, rather than gradually vanish along with its depth as the formula implies. This phenomenon may caused by the negative activation $A^{(N)}$ of the penultimate CONV layer, which lead to $g'(A^{(N)}) = \alpha = 0.01$, then the gradients are $2e^{-4} * 0.01 * h^{(N-1)} \approx 0$. Besides, on account of $h^{(N-1)} = \prod_{i=1}^{N-1} w^{(i)}$ (assume linear transformation), this phenomenon may also resulted by the tiny value of $h^{(N-1)}$. Secondly, in Figure 2(a), the first Conv layer of VGG08 has the largest gradient, which does not seem to be consistent with the previous conclusion. Given the expression of the first layer's gradient $\Delta w^{(1)} = \frac{\partial \text{Loss}}{\partial h^{(N)}} \prod_{i=1}^{N} g'(A^{(i)}) \prod_{i=1}^{N} w^{(i)} * X$, where X is the input image. Despite the benefit gained from BN layer, the large values of X may dominate the expression and give rise to this phenomenon.

Currently, the vanishing gradient problem has been clearly identified and relevant issues has been explained. **However, is vanishing gradient issue the only optimization problem that lead to VGG38 unexpectedly perform worse then the much shallower VGG08?** By visualizing the performance of the gradient-healthy VGG38 (shown in Figure 4), though the gradient flow of VGG38 has been repaired by BN technique, it still perform worse than VGG08 on training set. Besides its training accuracy 0.572 show a underfitting problem and obviously cannot much its capacity. This means there is something else prevents gradient-healthy VGG38 gain benefits from its depth, as metioned in introduction and (He et al., 2016), this phenomenon indicates the occurrence of the degradation problem. The original literature did not discover the reasons for the degeneration problem. One possible reason is that the correlation between gradients will decrease exponentially along with depth and lead to the white noise-like gradients, which are useless for training. (Balduzzi et al., 2017).

3. Background Literature

3.1. Batch Normalization

Internal Covariate Shift, which defined as the changing activation distribution caused by parameters update of preceding layers, cause a problem for each layer has to keep adapting the changing distribution of its activation. To mitigate this problem, Batch Normalization was proposed (Ioffe & Szegedy, 2015). Firstly, inspired by the idea of whitening, the study intended to do normalization on each layer's inputs. Then for reducing unaffordable computational complexity, two simplifications were made, that normalize each scalar feature independently rather than jointly and calculate its mean and variance on a mini-batch instead of on the whole training set. Besides, two learnable parameters were introduced for recovering the layer's representation ability affected by normalization. The study indicated that by using Batch Normalization the training of deep network can be greatly accelerated and can afford higher learning rate as well as need less weight penalty. Besides, it also benefit for solving vanishing gradient problem for the optimizer is less likely stuck in the saturated region of nonlinearity, and gradients are less depend on the scale of the parameters or of their initial values.

Experimentally, the benefits of Batch Normalization are obvious, but the reasons about why BN can promote training stay controversial. Recently, some works reveal that instead of mitigating Internal Covariate Shift, BN smoothed the objective function (Santurkar et al., 2019).

3.2. Deep Residual Learning for Image Recognition

Motivated by the counterintuitive phenomenon, that deeper network unexpectedly perform worse than the corresponding shallower network, the study conducted by (He et al., 2016) aims to solve this degradation problem. To this end, Residual Learning Framework was proposed, where each layer learns the residual mapping rather than unreferenced original mapping, for it was assumed easier to optimize. The authors stated residual mapping can only improve the capacity of layers for it can be degraded to original mapping by pushing residual to zero. Residual mapping was implemented by adding "shortcut connection", which will not add extra parameters when identity connection is added. Besides, they also provided three strategies to enable shortcut connection match the dimension, and the comparisons was well discussed. During the study, a series of experiments were presented, where residual learning framework successfully enabled deeper network gain accuracy from its depth. However, when network's depth exceeds 1000 layers the test accuracy decreased, and the authors believed this phenomenon was caused by overfitting.

This study fully discussed the details of experimental design and clearly explained the motivation behind their choices. However, it failed to identify the reason caused the degradation problem and it's beneficial effect for handling gradient vanishing problem are rarely explained. **It is worth mentioning that,** Batch Normalization were applied throughout the study, not only because vanishing gradient problem of baseline need to be fixed at first, but also for taking control experiments, which inspired our works.

3.3. Dense connected network

Inspired by the idea of "short connection", which can promote the information flow through the network, DesNet (Huang et al., 2017) was developed for enabling maximum information flow through the network. Similar but different with residual learning framework, DenseNet's each layer is connected to every subsequent layer. Different with ResNets where the features are summed together, DesNet choose to concatenate them. The authors stated that the inputs concatenation and the dense connection can create a map of collective knowledge, which can be accessed and reused by any following layers. One benefit of this is, there is no need for relearning redundant feature-map, and thus the DesNet need less parameter and less prone to overfit the data. Except the benefit of fewer parameter, better feature reuse, DesNet improves the conduction of information and gradient, which can solve gradient vanishing problem and promote the training. Finally, to evaluate the practicability of DesNet, DesNet were applied on four competitive object recognition tasks and achieve a great performance on all of them.

Comparisons: All the three approaches can handle vanishing gradient problems, but in two different ways. BN did it by decoupling the dependency between gradients and the scale of parameters, but ResNet and DesNet did it by adding short connections which can encourage the propagation of information and error signal. Only ResNet and DesNet can handle the degradation problem for the similar idea they started from. The key difference between ResNet and DesNet is the way they handle the identity mappings. ResNet sum them together $\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}$, but DesNet concatenate them $\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$. The concatenation maximally reserved the information and provide a collective knowledge map for the subsequent layers to access. DesNet also less likely overfit the data which provide a solution for the overfitting problem occured in ResNet. However, compared with BN and Residual Learning which can be easily integrated into existed network, DenseNet is less combinable for the relatively complicated architecture.

4. Solution Overview

4.1. Batch Normalization

As seen in section 3, Batch Normalization is beneficial for solving vanishing problem, and it can be easily integrated into the existed network. Besides, compared with the alternative that carefully initializing weights (Pascanu et al., 2013), adopting BN allow us concern less of the initialization and training procedures. Therefore, Batch Normalization is implemented at first to solve the gradient vanishing problem. **To be more specific,** before the activation going into ReLu, *BN* transformation was applied on each dimension of the layer's activation independently. Formulas of *BN* transformation are:

$$
\begin{aligned}
\mu_{\mathcal{B}} &\leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i & \sigma_{\mathcal{B}}^2 &\leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \\
\widehat{x_i} &\leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} & y_i &\leftarrow \gamma \widehat{x_i} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)
\end{aligned}
\tag{2}
$$

Where $\mathcal{B} = \{x_{1\ldots m}\}$ is the mini-batch that μ_β and σ_β calculated on, and $\{x_{1\ldots m}$ is the activations that *BN* applied on. γ and β are two parameters to be learned, which enable activation go into any part of the non-linear function. ϵ is a constant added for guarantee numerical stability. The formulas of backpropagation are:

$$\frac{\partial \ell}{\partial \widehat{x_i}} = \frac{\partial \ell}{\partial y_i} \cdot \gamma, \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x_i}} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} \left(\sigma_{\mathcal{B}}^2 + \epsilon\right)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x_i}} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x_i}} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x_i} \qquad \frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \tag{3}$$

But how does BN solve vanishing gradient problem? Mathematically, following the notation in original paper,where *BN* represents the *BN* operation, W represents the weights of current layer and u is the input of current layer. Following the definition of BN, $\mathrm{BN}(W\mathrm{u}) = \mathrm{BN}((aW)\mathrm{u})$ and then $\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial (aW)} = \frac{1}{a} \cdot \frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial W}$, which implies that, during backpropagation the gradient corresponding to the small W will be amplified. In another word, the gradient attenuation caused by small weights ($\prod_{i=n+1}^{N} w^{(i)}$ term in Equation 1) will be mitigated. Meanwhile, the formula also implies larger weights will obtain smaller gradients, thus the magnitude of weights can be well controlled, which not only can solve gradient exploding problem but also has beneficial effect on handling overfitting. Besides, the smaller weights will lead to smaller activation x_i and will lead to smaller σ_{β} further. Consequentially, larger $\frac{1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}}$ mulitiplied on $\frac{\partial \ell}{\partial x_i}$ will amplify the gradient.

4.2. Residual Learning Framework

As seen in section 3, Residual Learning Framework is targeted to solve degradation problem that hide behind the vanishing gradient issue. Therefore, after vanishing gradient problem been solved, Residual Learning Framework is integrated into existed network to sovle the remaining degradation issues. The residual block in this paper is shown in Figure 3, where $x_{n+1} = x_n + \mathcal{F}(x_n, W_n)$

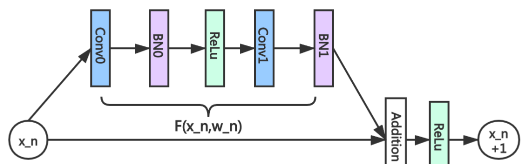


Figure 3. Residual Block Diagram

Then for a deeper N^{th} layer, the forward propagation is

$$x_N = x_n + \sum_{i=n}^{N-1} \mathcal{F}(x_i, W_i) \tag{4}$$

The backward propagation is:

$$\frac{\partial \varepsilon}{\partial x_n} = \frac{\partial \varepsilon}{\partial x_N} + \frac{\partial \varepsilon}{\partial x_N} \frac{\partial}{\partial x_n} \sum_{i=n}^{N-1} \mathcal{F}(x_i, W_i) \tag{5}$$

Note that for matching the dimension, a average pooling is adopted on x_n (when it need), for using 1×1, *stride* = 2 convolution kernel will introduce more parameters and more likely overfit the training data. Besides, stride equal to two will force the identity throw half of its features, which will cause lots of information loss.

But how does it solve degradation problem? The short connections enable layers to learn the residual mapping rather than fit unreferenced underlying mapping directly. Firstly, this architecture ensure the deeper network will not perform worse, because if the shallower model (a part of the deeper model) is optimal, the residual mapping of the above layers can be easily pushed to zero (for they have reference). Besides, the short connection can facilitate the flow of information through the network (Huang et al., 2017), which means the information sent by shallower layers will less likely to be lost due to the depth or smaller weights of deeper layers. This not only can accelerate the learning, but also can promote the propagation of error signal for the short connections are bidirectional.

5. Experiments

This section will apply the proposed solutions that Batch Normalization(BN) and Residual Learning Framework (RL) into practice. To be more specific, BN will be applied first, and then RL will be integrated into the network to solve the remaining degradation problem. The experiments were performed on CIFAR100 dataset (Krizhevsky et al., 2009), which is composed of 60,000 32×32 colour images, with 600 images contained each of the 100 classes. The training, validation and test sets sizes are 47500, 2500 and 10000 images respectively.

5.1. Solving Vanishing Gradient Problem

Whether BN can solve vanishing gradient issues and if its other benefits (such as can tolerate higher learning rate) can be exploit, as well as whether BN can completely solve the counterintuitive phenomenon mentioned above are the main focus of this experiment. In this experiment each activation has been batch normalized by adding a BN layer before the leakey Relu activation function. All the settings of VGG38 keep default that Adam optimizer, cosine learning schedule ($2e^{-5}$ minimum), 100 epochs, batch size=100, zero weight decay and 3 stages with 5 blocks per stage.

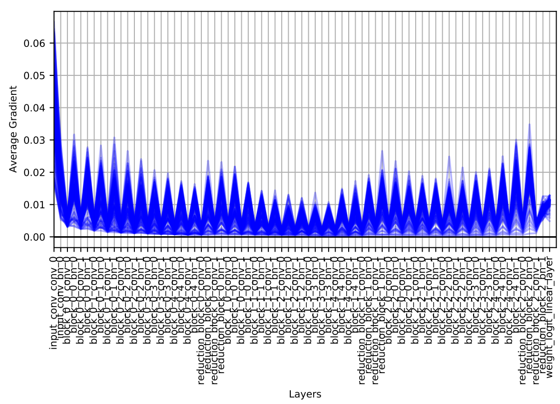
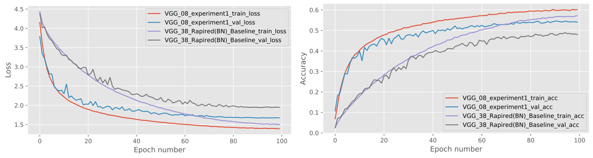


Figure 4. Gradient Flow of BN-based VGG38

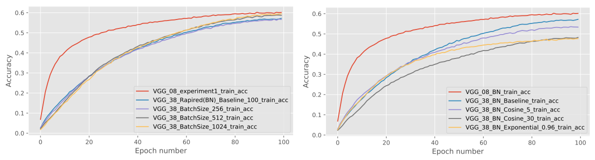
Note that for taking control experiment, BN is also applied on VGG08, and it will be unfair if just allowing VGG38 take advantage from a useful technique.



(a) Error Curves (b) Accuracy Curves

Figure 5. Comparison of BN applied VGG38 and VGG08

From Figure 3, 4 we can found that the gradient signal successfully flowed back to the input layer and VGG38 begin converging. However, the gradient-healthy VGG38 (0.572 training accuracy and 0.48 validation) still cannot beat the much shallower VGG08 (0.602 training accuracy, 0.541 validation accuracy) on training set, and the relatively stable tails of curves imply increasing epoch number is less likely to improve performance. *But what if this simply because the advantages of BN has not been fully exploited?*



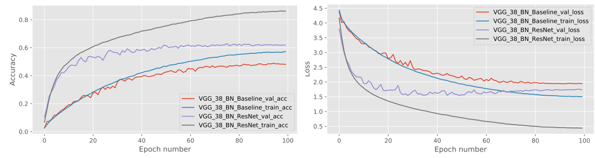
(a) Different Batch Size (b) Different Learning Rate

Figure 6. Training Accuracy against Parameter Setting

Since we noticed the simplification made in the original article, that calculate means and variances on mini-batch rather than on the whole training set, we first increased the batch size for larger batch size can provide better approximation and thus have better performance. According to Figure 6 (a), the training accuracy did increase with the batch size, but **the highest training accuracy** (0.595) achieved at 1024 batch size still worse than the baseline. We did not keep increasing batch size for the memory limit of CPU and the tiny increase of VGG38's training accuracy is not worth to explore further. **Interesting to mention, the running time decreased with the increase of batch size**. Secondly, as the original paper suggested, we increased the initial learning rate to 5 and 30 times of the defaulted $1e^{-3}$. However, as Figure 6(b) shown, the increased learning rate can only decrease the training performance. Besides, we also changed the learning schedule to exponential learning schedule (0.96 decay rate) to in accordance with the original paper. Still no clues show we should explore this aspect further. We didn't change the weight decay coefficient for BN itself can mitigate overfitting, and there is no sign of overfitting but underfitting. **Till now we can sure vanishing gradient is not the only problem that prevent VGG38 gain benefit from its depth, and the degradation problem is explicitly exposed.**

5.2. Solving Remaining Degradation Problem

Similar with the experimental procedures of (He et al., 2016), RL was implemented on the basis of the Batch Normalization. Without Batch Normalization, the semi-healthy VGG38 baseline cannot begin converging and thus cannot be compared with our experiment results. Following the design in original paper, a identity mapping from the second previous layer was added on the batch normalized output of current layer before going to leaky Relu function. Besides, all the setting still keep default, for we only care whether degradation problem can be solved in this subsection. The training performance comparisons of semi-healthy VGG38 (only use BN) and healthy VGG38 (BN+ResNet) is shown below:



(a) Accuracy Curves (b) Error Curves

Figure 7. Performances of Semi-healthy and Healthy VGG38

NETWORK	TRA_{acc}	TRA_{err}	VAL_{acc}	VAL_{err}
VGG08-BN	0.602	1.392	0.541	1.673
VGG38-BN	0.572	1.503	0.48	1.947
VGG38-BN+RES	0.862	0.438	0.619	1.739

Table 1. Accuracies Errors of semi-healthy and healthy VGG38

As shown in Figure 7(a) and Table 1, with the help of Residual Learning Framework the healthy VGG38 easily exceeded both BN-applied VGG08 and the semi-healthy VGG38 by about 29 percent training accuracy. For exploring how Residual Learning affected gradient signal, the gradient flow of healthy-VGG38 was plotted.

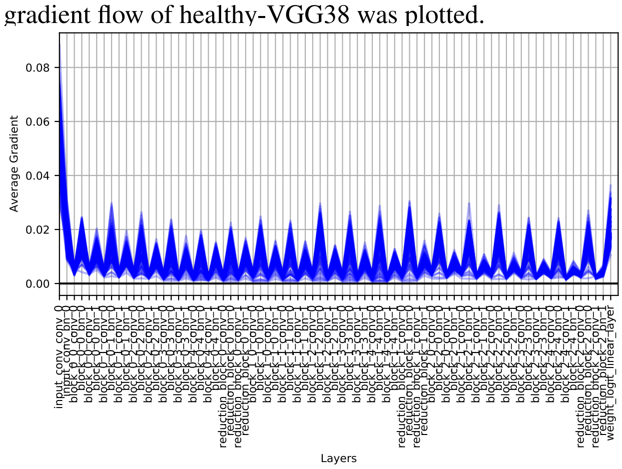


Figure 8. Gradient Flow of BN Assembled VGG38

5.3. Solving Overfitting and Testing Stability

Till now the vanishing gradient problem and degradation problem have been well settled. This experiment aims to solve the overfitting problem exposed in Figure 7(b), for achieving a better and more stable performance. Instead of using dropout we increased the weight decay coefficient, because dropout keep changing the network architecture for each batch and thus a little bit risky. The coefficient was adjusted from $1e-4$, for the original paper (He et al.,

2016) adopted this value on VGG network and gained great performances. The searching stopped at $1e-2$ for the validation accuracy curve not stable anymore.

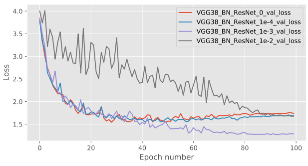


Figure 9. Validation Error

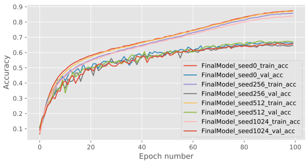


Figure 10. Test Stability

As shown in Figure 9, when coefficient equal to $1e^{-3}$ the overfitting problem disappeared, thus we adopted this coefficient and run our final model on multiple seed to test its stability (results shown in Figure 10 and Table 2).

SEED	TRA_{acc}	TRA_{err}	VAL_{acc}	VAL_{err}	TEST_{err}	TEST_{err}
0	0.874	0.435	0.663	1.286	0.655	1.312
256	0.859	0.487	0.648	1.343	0.646	1.360
512	0.871	0.451	0.670	1.273	0.657	1.321
1024	0.838	0.552	0.658	1.300	0.647	1.316

Table 2. Accuracies Errors of semi-healthy and healthy VGG38

According to Table 2, **the final test accuracy** is reported as 0.651 ± 0.0048 (*mean* ± *std*), and loss is 1.327 ± 0.0191. We did not do fine-grained searching and other hyperparameters searching further, for our purpose is not provide a model with best performance, but solving the identified problem and let model achieve a stable performance.

6. Discussion

Firstly, compared with Figure 2, the gradients shown in Figure 3 successfully flowed back to the input layer. However, the peaky shape of the gradient flow is unexpected in advance. Following the notation in section 4.1, where *BN* represents the *BN* transformation which applied on each dimension of $A^{(n)} = W^{(n)}h^{(n-1)}$ independently, $A^{(n)}$ and $h^{(n-1)} = g(\text{BN}(A^{(n-1)})$ are current layer's activation and input respectively. According the definition of BN, $\text{BN}(A^{(n)}) = \text{BN}((aA^{(n)}))$, where a is a scalar. Then $\frac{\partial \text{BN}(a(A^{(n)}))}{\partial (A^{(n)}} = \frac{1}{a} \cdot \frac{\partial \text{BN}(A^{(n)})}{\partial A^{(n)}}$, which means smaller weights will lead to smaller a and then gradients of current layer's activation will be amplified when gradient signal flow to it. **The activation's gradient is just the gradients shown on 'bn' layers**, so the gradient flow **show a upward trend on 'bn' layers**. However, when the gradient signal flow to the $W^{(n)}$, a tiny value $h^{(n-1)}$ will be multiplied on it, thus the gradient flow **show a downward trend on CONV layers**. The healthy gradient flow and the performance curves shown in Figure 4, indicates BN enabled broken VGG38 to converge and **solved the vanishing gradient problem effectively**. However, its training accuracy 0.572 shows a underfitting problem and obviously cannot match its expected fitting capacity. Besides the experimental results given in Figure 5 shows that reasonably changing hypermeters still cannot enable semi-healthy VGG38 gain accuracy from its depth and exceed baseline in terms of training accuracy, which make the degradation problem more explicit. Run time decrease with the increase of batch size may because larger batch size will lead to less mean and variation calculation. Meanwhile, contrary to the original literature (Ioffe & Szegedy, 2015), increasing learning rate can only make things worse. This may partly due to the experiments were run on different datasets for different tasks, but more likely because the degradation problem prevented the network from gaining benefit from BN further.

The huge training accuracy improvement (29% shown in Figure 6(a) and Table 1indicates Residual Learning Framework empowered the semi-healthy VGG38 to reap benefits from the extra layers and beat the baseline easily. Besides, the overfitting phenomenon shown in Figure 6(b) also indicates the fitting ability of semi-healthy VGG38 has been recovered successfully. Therefore the **Residual Learning Framework successfully handled the degradation problem**. However, the experiments conducted by (He et al., 2016) on the CIFAR-10 dataset, shows the overfitting problem occurs only when the depth of network exceeds 110 layers. This may because our training images for each class is 10 times smaller then their, for we classified the training images into 100 classes rather than their 10 classes, thus our VGG38 may unnecessary deep for the smaller dataset.

There is an interesting phenomenon discovered in Figure 8. Compared with the gradient flow of semi-healthy VGG38, after applying Residual Learning Framework further the gradient of 'bn1' where identity mapping was added has reduced.This because 'bn1' actually learnt the residual mapping. Compared with learning underlying mapping directly, learning residual mapping need weights of this layer change less for it has a identity mapping as a reference. Consequently, the layer will have smaller gradient for its weights contribute less to the output. Besides, as the depth increasing the reference should be more and more accurate for the increasing abstraction power provided by preceding stacked layers, and thus the weights of deeper layer will change less. Indeed we can found this trend on the Figure, where the 'bn1' near the input has larger gradient and the 'bn1' near the output has smaller gradient.

7. Conclusions

In this report we explored the optimization problem of deep CNN training that vanishing gradient issue and degradation problem. After applying Batch Normalization and integrating Residual Learning Framework further, we successfully addressed these problems. Finally, after solving overfitting, We achieved a reasonble training accuracy that 0.861 ± 0.0142 and stable test accuracy that 0.651 ± 0.0048. During this procedure we not only have a deeper understanding of these optimization problem, we also learned how to apply the learned knowledge into practice. Besides, we also learned it is better to start from some simple baselines and do experiment incrementally, for it may expose more hidden problems, such as the degradation problem found in this report. In the future we would like to explore the causes of the degradation problem and the real principle of Batch Normalization for it stay controversial.

References

Balduzzi, David, Frean, Marcus, Leary, Lennox, Lewis, J. P., Ma, Kurt Wan-Duo, and McWilliams, Brian. The shattered gradients problem: If resnets are the answer, then what is the question? *CoRR*, abs/1702.08591, 2017. URL http://arxiv.org/abs/1702.08591.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE, 2016. ISBN 9781467388511.

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 (CVPR)*, July 2017.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

Krizhevsky, Alex et al. Learning multiple layers of features from tiny images. 2009.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318, 2013.

Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, and Madry, Aleksander. How does batch normalization help optimization?, 2019.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. 2014.