MLP Coursework 2: Investigating the Optimization of Convolutional Networks

s2014864

Abstract

In this work, we researched the different neural network, VGG_38 and VGG_08's performance on CIFAR100 datasets. Deep neural network is difficult to optimize, due to the linear transformation and nonlinear activation mapping in each layer, the parameters change in underlying network will be amplified as the network layers deepening, which will result the vanishing gradient problem. In our work, VGG_38 performs worse compared with VGG_08 on CIFAR100 datasets and was difficult to optimize as it met the vanishing gradient problem. To solve it, three optimizing method and their related papers were reviewed in our work. We used one of the method, Batch Normalization, with hyper-parameters adjustment, it improved the classification test accuracy for VGG_38 on CIFAR100 from 1% to 53%

1. Introduction

In recent years, with the remarkable development of deep learning technology, especially convolutional neural network (CNN), promising performance has been observed in various computer vision and machine learning tasks[9]. The layers of neural networks are getting deeper and deeper, and dozens or even hundreds of layers of neural networks have appeared now, and the results obtained are much better than those of shallow neural networks. Deeper network takes the advantage of higher degree of abstraction of features and the more parameters that can be adjusted, which results in a better fitting effect. But constantly deepening the neural network will bring some negative problems: Vanishing gradient problem, Overfitting and Degradation[2].

Based on the performance for neural networks with different network depth, VGG_08 and VGG_38, on CIFAR100. Two problems were identified: Vanishing gradient problem and internal covariate shift problem, which are also the main research work in this paper. The reason for this problem is that the upper layer network needs to be constantly adjusted to adapt to changes in the distribution of input data, resulting in a decrease in network learning speed. As the number of network layers deepens, the training process of the network easily falls into the gradient saturation problem, which even may make the network can not converge.

Batch Normalization[1] was one useful method to reduce the above problem influence. We tried to reproduce and imitate the method (Ioffe Szegedy, 2015) applied. Adding Batch Normalization after every convolution layer and before activation function ReLU, the results were in line with our expectations, the accuracy for VGG_38 improved from about 1% to 50%. Rethinking another network structure for adding Batch Normalization proposed by (G. Chen et al.,2019)[7]. Another structure was applied for our model, adding Batch Normalization after ReLU, the experiment results represented this method effectively improved VGG_38's performance to some extent. After above steps, the training accuracy were higher than test accuracy about 10%, our later work were aimed at finding the effective hyper-parameters to improve the generalization ability of our model.

To summarize, we list our main contributions as follows:

• We analyzed the vanishing gradient and internal covariate shift problem in deeper network VGG_38. To solve the problems, we applied the technique Batch Normalization to optimize our neural network. We applied another method based on (G. Chen et al.,2019)'s theory, which further optimize our neural network. We tried different valid hyper-parameters to improve our model's generalization ability.

• To corroborate our theoretical analysis, we conduct experiments on CIFAR100 datasets. The results verified that our implementations above improved the classification performance with higher accuracy and faster converge speed.

2. Identifying training problems of a deep CNN

• Identifying training problems:

To identify the training problems of a deep CNN. Two deep CNNs, VGG_08 and VGG_38, will be adopted. We will test their performance on the dataset CIFAR100.

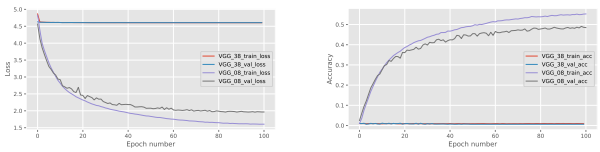

Figure 1. Training and validation plots of Healthy (VGG_08) vs Broken (VGG_38) in terms of error and accuracy

The resulting performance these two networks are so different. Compared with the healthy 7 layer CNN(VGG_08), the deeper network 37 layer CNN(VGG_38) performs bad as it has about 0 accuracy and unable to converge to a low error.

Before analyzing the reasons for the bad performance of VGG_38. One thing we concerned is the difference between VGG_38 and VGG_08. There is no essential difference between the two, but the depth of the network is different.

We do the hypothesis for the issue's reasons in Figure1. The first hypothesis we thought is the dataset issues. Whether it loads the wrong dataset to feed the VGG_38 network, whether it mixes the datasets features etc. Another hypothesis is based on the analysis in above paragraph, the issue comes from the model difference-network depth. As the number of network layers deepens, the expressive and abstract capabilities of the network will also increase. For two network structures with the same time complexity, the deeper network performance will be improved[4]. But some negative problems will be raised with deepening the neural network: vanishing/exploding gradients[5], overfitting and degradation[2]. excluding the overfitting hypothesis(The accuracy for training and testing data are both 0), the left two also may be the issue's reasons.

• **Quantitative Analysis of the problem:**

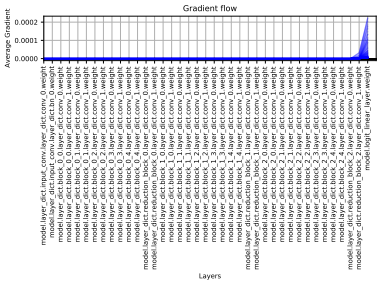

Figure 2: Gradient Flow in each layer for the VGG_38 network

In figure 2, it shows the gradients with respect to the weights of the model in each layer. The average gradients for each layer in VGG_38 network are almost about 0, which will affect the loss of the network to be the constant without change. Furthermore, this is the **'Vanishing gradient phenomenon'[8]**, which is often encountered when training artificial neural networks with gradient-based learning methods and backpropagation, resulting the weight can not be effectively updated, or even the neural network may be unable to continue the training. Hence, the loss curves for VGG_38 is a straight line in Figure 1. As the loss does not change, it will result the 37 layers CNN can not converge and train. From the quantitative observation of the broken network, it supports the hypothesises of vanishing/exploding gradients problem and degradation, improves our confidence on the two hypothesises.

3. Background Literature

In this section, three research papers will be reviewed. We will discuss the problems these papers address, method they adapt to solve problems, the difference and relations between these papers and problems raised in section 2 and brief conclusions for different papers.

• **Batch normalization: Accelerating deep network training by reducing internal covariate shift[1]**
As the training progresses, the parameters in the network are constantly updated as the gradient drops. On the one hand, when the parameters in the underlying network change slightly, due to the linear transformation and nonlinear activation mapping in each layer, these weak changes will be amplified as the number of network layers deepen. On the other hand, changes in parameters cause the input distribution of each layer to change, and the upper layer network needs to constantly adapt to these distribution changes. This is the **internal covariate shift** in this paper

To reduce the ICS(internal covariate shift) and improve training, this paper first introduces 'whitening' method to fix the distribution of the layer input. But this method has some disadvantages, one is costly, another is not everywhere differentiable. Then, the 'Batch Normalization' is introduced. It makes simplifications and improvement based on 'whitening'. But normalizing the inputs of sigmoid or tanh will constrain them to the linear regime of the nonlinearity and decrease the data expression ability to some extent. To solve it, Batch Normalization introduces a pair of parameters and to scale and shift the normalized value to retain the distribution information of the original input parameters. At last, Batch Normalization improve the network performance and reduce the ICS with sufficient experiments support.

Based on the experiments and analysis in this paper, it can conclude that Batch Normalization is a useful method to solve the problem of internal covariate shift. What's more, the problem of ICS will be more serious as the number of network layers deepens. The over-number of network layers for our model may results in the problem similar with the ICS 's problem.

• **Deep residual learning for image recognition[2]**
This paper introduces residual learning principle, which allows the network depth to be increased deeper without causing degradation problem. Increasing depth of network is helpful to improve accuracy, but it will take several problems:
• vanishing/exploding gradients problem (largely addressed by normalized initialization and intermediate normalization layers)
• degradation problem

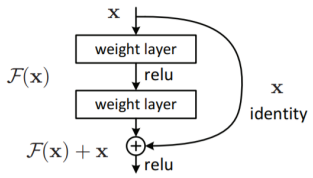

Figure 3: Residual learning: a building block[2]

Deep residual learning tries to make a certain transformation of the network units to improve the degradation problem. In this paper, it tries to make each few stacked layers fit a 'residual mapping' $\mathcal{F}(x)$, not directly fit a desired underlying mapping $\mathcal{H}(x)$, where $\mathcal{H}(x)=\mathcal{F}(x)+x$. The stacked nonlinear layers fit the mapping of $\mathcal{F}(x)$, adding the 'shortcut connections' to perform identity mapping x. Then, using the relu activation to the adding outputs $\mathcal{F}(x)+x$. The entire residual network can be easily implemented with the mainstream libraries and trained by SGD with backpropagation.

This paper introduces the problems deeper neural networks will result in. In response to these problems, it proposed the methods from other papers, and proposed the deep residual learning for the specific problem, degradation problem. In conclusion, after applying the residual learning, the deep network is easier to optimize and get the lower error.

The degradation problem proposed in this paper: with the network depth increasing, accuracy get saturated and then degrades rapidly(both training and test accuracy), which is similar with our problem. The deeper network, VGG38 , its train and test accuracy both drops to about 0 compared with the VGG08.

• **Deeply-supervised nets[3]**
In this paper, it raises a problem as CNNs become increasingly deep: the input information or gradient will gradually disappear and be 'washed' away as it passes between many layers. To solve this problem, it proposes an architecture with a simple connectivity pattern, they connect all layers that match the size of the feature map directly with each other. Each layer takes additional inputs from all previous layers and uses its own feature maps as input for all subsequent layers.

DenseNet introduces direct connections between any two layers with the same feature-map size. With sufficient experiments results, DenseNet performs well with less computation and fewer parameters. DenseNets may also be good feature extractors for various computer vision tasks that build on convolutional features. The problem in this paper is caused by the increasingly deepening CNNs, which is similar with our the cause of our problem to some extent.

The common character of the three papers is they are trying to solve the problems caused by neural network layers increasingly deepening or improve the performance of the deep neural network, which are related with the problem raised in section 2. Based on this, we will go further to investigate whether it may helps to solve our problem.

4. Solution Overview

In this paper, we decided to choose the solution Batch Normalization[1] to solve our problem. S. Ioffe and C. Szegedy states the specific phenomenon **internal covariate shift**[1] , which will cause:
• The upper network needs constant adjustment to adapt to the change of input data distribution, which leads to the decrease of network learning speed
• The training process of the network is easy to fall into the gradient saturation problem with the number of network layers deepening, which slows down the network convergence speed

Furthermore, the problem of internal covariate shift will be more serious as the networks deepening. and the relation with our problem are:
• In figure 1, the loss curve of VGG_38 can be seen as a staright line with very low gradient. The loss has barely changed as epoches increasing. As discussed about **internal covariate shift problems**, the network learning speed will be very low if the networks layers is so deep. Furthermore, we can see that the average gradient value in most layers approximately equal to 0 in figure 2. Hence,we make the assumption that the previous problems are caused by the internal covariate shift, resulting into the VGG_38 network has a very low learning speed.
• The internal covariate shift slows down the network convergence speed, make our network hard to converge. In figure 1, the deeper network, VGG_38 is hard to converge, where its loss keeps high with the epoches increasing.

To deal with internal covariate shift problem, Batch Normalization was applied. It achieve two main functions:
• Reduce drastic changes in network parameters caused by changes in the distribution of data input to the network layer, and accelerate network convergence.
• Reduce the sensitivity of the network to large-scale parameter changes in the reverse calculation process, and make the network parameter update more stable.

Batch Normalization first normalized every independent feature,We consider a batch training, pass in m training samples, and focus on a certain layer in the network:

$Z \in R^{d_l \times m}$

We pay attention to the j_{th} dimension of the current layer, that is, the j_{th} neuron node, then there is $Z \in R^{1\times m}$. Our current dimensions are normalized:

$u_j = \frac{1}{m}\sum_{i=1}^{m} Z_j^{(i)}$	batch mean
$\delta_j^2 = \frac{1}{m}\sum_{i=1}^{m}(Z_j^i - u_j)^2$	batch variance
$\hat{Z}_j = \frac{Z_j - u_j}{\sqrt{\delta_j^2+\epsilon}}$	normalization
$y_j = \gamma\hat{Z}_j + \beta$	scale and shift

From the above steps, Batch Normalization simplified the calculation process by normalize the input data on the one

hand, one the other hand, the standardized data retained the original expression ability by adding two learnable parameters γ and β.

From the analysis above, we do the hypothesis that the bad performance of VGG_38 network was caused by internal covariate shift and vanishing gradient problems. Based on above analysis about Batch Normalization, we decided to choose the Batch Normalization as our solution to solve the problems.

5. Experiments

To verify the effects of Batch Normalization to our VGG_38 network. We considered the classification tasks on CIFAR100 dataset to test the performance of our neural network-VGG_38. CIFAR100 consists of 60,000 32 × 32 colour images in 100 classes, with 600 images per class. The 100 classes in the CIFAR100 are grouped into 20 superclasses.Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs)[6].In our experiments, there are 47500 images for training sets, 2500 images for validation sets and 10000 images for test sets. In the experiments part,we will use the broken network,VGG_38, consisting of 37 convolutional layers and one fully connected layer, and the CIFAR100 input dataset as our experiment foundation.

To **reduce the instability** of our experiments' result, we trained our network in **different seeds** in below experiments and observed whether they varied a lot, if the results are similar, we will use the experiments for seed 0 to keep all experiments results based on the same seed.

5.1. Adding Batch Normalization

To find whether the Batch Normalization could solve our problem. Based on the S. Ioffe's Batch Normalization experiments, we added batch normalization after every convolution layers in VGG_38 and compared its performance with VGG_38 without Batch Normalization.

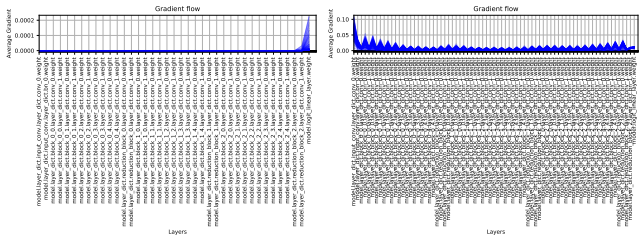

Figure 4: Gradient Flow in each layer for the VGG_38 network without adding BN and adding BN

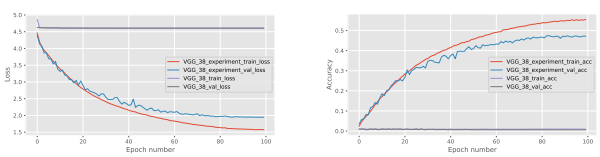

Figure 5. Training and validation plots of VGG_38 adding BN and without adding BN in terms of error and accuracy

We did not change any other parameters of the VGG_38 network but only added the Batch Normalization after every convolution layer. After adding Batch Normalization, the performance of VGG_38 improved a lot. From figure 4, it can be seen that the vanishing gradient problem has been solved to some extent. And the training and val curve converged around about 80 epoches. The test accuracy increased from 1% to 47.43%, and the test loss decreased from 4.609 to 1.968. Batch Normalization did really improve our network, however, the network performance was not good enough, in later section, we would try to find methods to improve further the VGG_38 performance.

5.2. Batch Normalization Before or After ReLU

In S. Ioffe and C. Szegedy's Batch Normalization experiments[1], they added the Batch Normalization **before the activation function**. However, a different point of view thinks that we should not place Batch Normalization before ReLU since the nonnegative responses of ReLU will make the weight layer updated in a suboptimal way. (G. Chen et al.,2019) In this section, we will test and compare the performance of the neural network VGG_38 when adding Batch Normalization before the activation function ReLU or after the activation function.

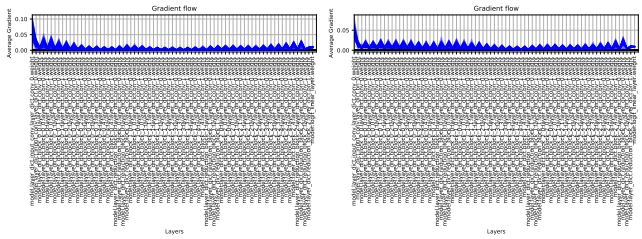

Figure 6: Gradient Flow in each layer for the VGG_38 network adding BN before ReLU and adding BN after ReLU

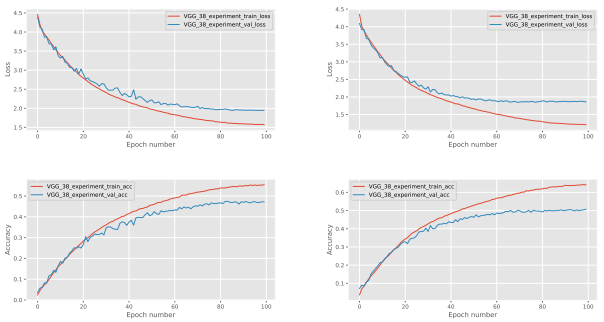

Figure 7. Training and validation plots of VGG_38 adding BN before ReLU(two figures on the left) and adding BN after ReLU(two figures on the right) in terms of error and accuracy

	train acc	val acc	test acc	train loss	val loss	test loss
BN before ReLU	55.5%	47.2%	47.4%	1.571	1.952	1.968
BN after ReLU	64.1%	50.7%	51.9%	1.211	1.861	1.852

Table 1. BN before ReLU and after ReLU networks' accuracy and loss performance on CIFAR100 datasets

In above table. It can be seen that BN after ReLU has higher accuracy on train and test datasets and lower loss compared with adding BN before ReLU. Based on the experiments' results, adding BN after ReLU truly improved our network performance compared with adding BN before ReLU, which verified (G. Chen et al.,2019)'s perspective

5.3. hyper-parameters searching

In this section, valid hyper-parameter search will be implemented.

weight decay coefficients: From previous experiment in section 5.2, we observed that for the model adding BN after ReLU, the training accuracy is 64.1% and the test accuracy is 51.9%. Weight decay can be seen as a sample regularizer to prevent overfitting and increase the generalization performance of the model(Leslie N. Smith,2018). To improve its generalization ability, we will change the weight decay based on the model of adding BN after ReLU.

	train acc	val acc	test acc	train loss	val loss	test loss
decay coefficient=0	64.1%	50.7%	51.9%	1.211	1.861	1.852
decay coefficient=0.001	62.9%	54.1%	52.7%	1.278	1.692	1.707
decay coefficient=0.005	46.7%	42.4 %	43.0%	1.963	2.084	2.089

Table 2. different weight decay coefficient settings for networks' accuracy and loss performance on CIFAR100 datasets

Compared with the network with default weight decay coefficient 0, the network with coefficient 0.001 has lower training accuracy and higher test accuracy. However, for the network with coefficient 0.005, it performed bad as its lower accuracy and higher loss. We decided to choose decay coefficient 0.001 as it can improve the network's generalization performance to a certain extent.

Dropout: Dropout (Srivastava et al., 2014) is typically used to reduce overfitting and improve the generalization of model. We tried to add the Dropout after every Batch Normalization, but the results were not positively improve our model's performance. This experiment section based on the network of adding BN after ReLU.

	train acc	val acc	test acc	train loss	val loss	test loss
Without Dropout	64.1%	50.7%	51.9%	1.211	1.861	1.852
Dropout(p=0.5)	0.7%	0.5%	0.5%	4.608	4.610	4.611
Dropout(p=0.2)	0.9%	0.6%	0.5%	4.605	4.607	4.610

Table 3. Adding Dropout with different setting for networks' accuracy and loss performance on CIFAR100 datasets

After adding the Dropout layers, our network performed very bad. (S.Ioffe C.Szegedy,2015) proposed that in a batch-normalized network, dropout could be removed or reduced. (P. Luo et.al,2019[14)] proposed that Batch Normalization can be viewed as an implicit regularizer. Based on their perspectives, we thought combining Batch Normalization and Dropout may result in a kind of over-regularizing. (Xiang Li et.al,2019[15]) proposed that above problem were caused by **Variance Shift**.When we transfer the state of a particular neural unit from training to testing, Dropout will change the variance of that neural unit. However, BN will maintain its statistical variance, which is accumulated from the entire learning process, during the testing phase. When Dropout is applied before BN, the inconsistency of the variance (we named this scheme "variance shift") leads to unstable numerical behavior in the inference, which ultimately leads to more false predictions.

Due to the bad performance and above analysis, we decided to remove dropout to improve our model's generalization ability.

Batch Size:(Keskar N S et.al,2016) proposed that the large batchSize converges to the sharp minimum, and the small Batchsize converges to the flat minimum, which has better generalization ability. We tried to decrease the batch size to improve our model's generalization ability.

	train acc	val acc	test acc	train loss	val loss	test loss
Batch size=100	64.1%	50.7%	51.9%	1.211	1.861	1.852
Batch size=80	61.7%	51.3%	51.5%	1.318	1.843	1.850
Batch size=50	62.1%	51.6%	52.2%	1.305	1.811	1.785

Table 4. Different batch size for networks' accuracy and loss performance on CIFAR100 datasets

We observed that decreasing batch size, the network has lower training accuracy and higher test accuracy, which implied that its generalization ability improved. After this, we tried to combine these method to see whether it could improve our model's generalization ability further.

combine batch size and weight decay: From above analysis, we saw that the weight decay and batch size change could improve the model's generalization ability to different certain extent. So, we decided to combine them in this section.

	train acc	val acc	test acc	train loss	val loss	test loss
Default setting	64.1%	50.7%	51.9%	1.211	1.861	1.852
Weitght decay=0.001,Batch size=50	53.1%	49.4%	50.0%	1.696	1.853	1.828
Weitght decay=0.001,Batch size=80	59.4%	52.0%	51.7%	1.435	1.741	1.777

Table 5. Different combination of batch size and weight decay for networks' accuracy and loss performance on CIFAR100 datasets

We found that the combination of batch size and weight decay did not meet our expectations when decay coefficient=0.001 and Batch size=50. The test accuracy was lower than the default setting. The reasons we thought that as the batch size and weight decay both have regularization function. Combining them could be seen as adding the regularization ability, where we could see the training accuracy decreased more than applying each of them separately to our model.

From the above experimental results and analysis, we de-

cided to use weight decay coefficient =0.001 and adding BN after ReLU to improve our network.

6. Discussion

The comparison shown in Figure 1 and Figure 2 supported our hypothesis for VGG_38's vanishing gradient problem. It can be found that in every network layer, the gradient values are all approximately equal to 0. From the analysis about the reasons for vanishing gradient and internal covariate shift, we could find the learning speed was almost decreased to 0 and the network VGG_38 could not converge.

Similar with problem and solution in (S.IoffeC.Szegedy, 2015), we followed the steps to add the Batch Normalization after every convolution layer and before activation function ReLU. From the experimental comparison results shown in Figure 4 and Figure 5, we observed that after applying Batch Normalization to our network, the gradient values were not 0 in every network layer and from the accuracy and loss curves, the network curves successfully and stably converged. This is because Batch Normalization fixed the distribution of network input values at each layer to alleviate internal covariate shift problem. After changing the Batch Normalization from before ReLU to after ReLU, from figure 7 and table 1, we found that it had positive effect on our network as the improved accuracy and decreased loss. The reason is that the nonnegative responses of ReLU will make the weight layer updated in a suboptimal way(G. Chen et al.,2019). We observed that in figure 8 and table 2, the valid hyper-parameters change improved our model's generalization ability, where it had lower gap between train and test accuracy, also higher test accuracy and lower test loss, as discussed in experiments section, every hyper-parameter change had its unique function, we did not re-discuss its meaning here.

7. Conclusions

In this paper, we explore the optimization problems in training deep CNN architectures due to vanishing gradient and internal covariate shift problems, and we show the benefits of the Batch Normalization to this problem. What's more, we rethink the usage of Batch Normalization different with (S.IoffeC.Szegedy, 2015), and find that Batch Normalization transformation's position influence our model's performance deeply. As shown in our theoretical analysis and experiment results, the Batch Normalization improves deep CNNs' performance and solve the optimization problems, we achieve faster convergence speed, more stable and useful training process and better generalization performance. In future, we would like to gain further insights into more advanced normalization methods such as group normalization (Wu He, 2018) and better gradient-based training methods(Bengio, Y,2012) into our CNNs layers to achieve better performance

8. Reference

[1] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[2] He, Kaiming, et al. "Deep residual learning for image recognition." arXiv preprint arXiv:1512.03385 ,2015.

[3] G. Huang, Z. Liu, K. Q. Weinberger, and L. Maaten. Densely connected convolutional networks. In CVPR, 2017.

[4] He, Kaiming, and Jian Sun. "Convolutional neural networks at constrained time cost." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[5]X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.

[6]"CIFAR-10 and CIFAR-100 datasets", Cs.toronto.edu. [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html. [Accessed: 25- Nov- 2020].

[7]G. Chen, P. Chen, Y. Shi, C. Hsieh, B. Liao, and S. Zhang, "Rethinking the usage of batch normalization and dropout in the training of deep neural networks," arXiv preprint arXiv:1905.05928, 2019

[8]Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(2):107–116, 1998.

[9]Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS), 2012.

[10]Wu, Y. and He, K. Group normalization. arXiv preprint arXiv:1803.08494, 2018.

[11]Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures, in: Neural Networks: Tricks of the Trade. Springer, pp. 437–478.

[12]Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820,2018.

[13]Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014.

[14]P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," in International Conference on Learning Representations (ICLR), 2019.

[15]Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch nor-

malization by variance shift. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2682–2690, 2019.

[16]Keskar N S, Mudigere D, Nocedal J, et al. On large-batch training for deep learning: Generalization gap and sharp minima[J]. arXiv preprint arXiv:1609.04836, 2016.

[17]Smith S L, Kindermans P J, Ying C, et al. Don't decay the learning rate, increase the batch size[J]. arXiv preprint arXiv:1711.00489, 2017.