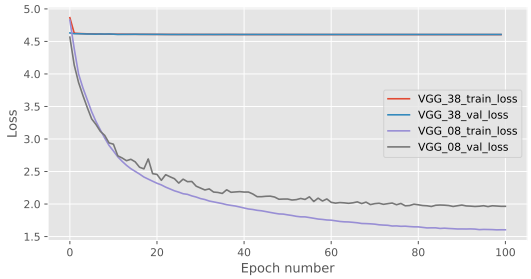
MLP Coursework 2

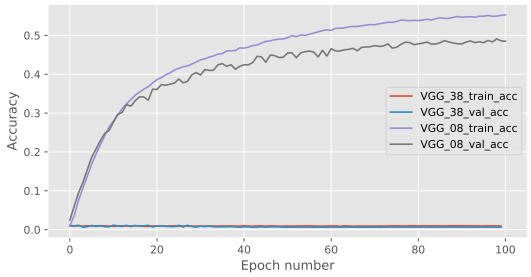
Xiaoyu Jiang

Abstract

Deep neural networks have become the state-of-the-art in many standard computer vision problems thanks to more powerful neural networks and large labeled datasets. While very deep networks allow for better deciphering of the complex patterns in the data, training these models successfully is a challenging task due to problematic gradient flow through the layers, known as vanishing/exploding gradient problem (VGP and EGP respectively). In this report, we first analyze this problem in VGG models with 8 and 38 hidden layers on the CIFAR100 image dataset, by monitoring the gradient flow during training. We explore known solutions to this problem including batch normalization or residual connections, and explain their theory and implementation details. Our experiments show that batch normalization and residual connections effectively address the aforementioned problem and hence enable a deeper model to outperform shallower ones in the same experimental setup.



(a) Loss per epoch



(b) Accuracy per epoch

Figure 1. Training curves for VGG08 and VGG38

1. Introduction

Despite the remarkable progress of deep neural networks in image classification problems (Simonyan & Zisserman, 2014; He et al., 2016), training very deep networks is a challenging procedure. One of the major problems is the VGP, a phenomenon where gradients from the loss function shrink to zero as they backpropagate to earlier layers, hence preventing the network from updating its weights effectively. This phenomenon is prevalent and has been extensively studied in various deep network including feed-forward 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 occurred in the VGG38 model and addressing it by implementing two standard solutions. In particular, we first study the "broken" network in terms of its gradient flow, norm of gradients with respect to model weights for each layer and contrast it to ones in the healthy VGG08 to pinpoint the problem. Next, we review two standard solutions for this problem, batch normalization (BN) (Ioffe & Szegedy, 2015) and residual connections (RC) (He et al., 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 tackle the vanishing/exploding gradient problem, therefore enabling the training of the VGG38 model, the best results are obtained by combining both BN and RC.

2. Identifying training problems of a deep CNN

Concretely, training deep neural typically involves three steps, forward pass, backward pass (or backpropagation algorithm (Rumelhart et al., 1986)) and weight update. The first step involves passing the input x^0 to the network and producing the network prediction and also the error value. In detail, each layer takes in the output of the previous layer

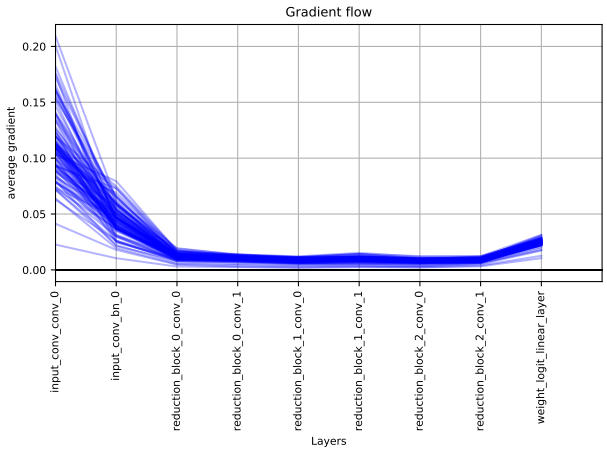


Figure 2. Gradient flow on VGG08

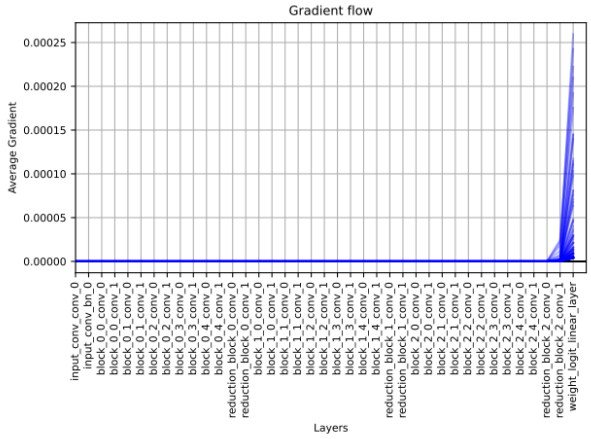


Figure 3. Gradient Flow on VGG38

and applies a non-linear transformation:

$$\boldsymbol{x}^{(l)} = f^{(l)}(\boldsymbol{x}^{(l-1)}; W^{(l)}) \tag{1}$$

where (l) denotes the l-th layer in L layer deep network, $f^{(l)}(\cdot, W^{(l)})$ is a non-linear transformation for layer l, and $W^{(l)}$) are the weights of layer l. For instance, $f^{(l)}$ is typically a convolution operation followed by an activation function in convolutional neural networks. The second step involves the backpropagation algorithm, where we calculate the gradient of an error function E (e.g. cross-entropy) for each layer's weight as follows:

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

This step includes consecutive tensor multiplications between multiple partial derivative terms. The final step involves updating model weights by using the computed $\frac{\partial E}{\partial W^{(l)}}$ with an update rule. The exact update rule depends on the optimizer.

A notorious problem for training deep neural networks is the vanishing/exploding gradient problem (Bengio et al., 1993) that typically occurs in the backpropagation step when some of partial gradient terms in Eq. 2 includes values larger or smaller than 1. In this case, due to the multiple consecutive multiplications, the gradients w.r.t. weights can get exponentially very small (close to 0) or very large (close to infinity) and prevent effective learning of network weights.

Figures 2 and 3 depict the gradient flows through VGG architectures (Simonyan & Zisserman, 2014) with 8 and 38 layers respectively, trained and evaluated for a total of 100 epochs on the CIFAR100 dataset. Comparing Figure 2 and 3, it is obviously that VGG38 suffers severe vanishing gradient problem(VGP), gradients for most layers become extremely close to zero, although there is some recovery in reduction_block_2_conv1 and conv2,which only appropriate about 0.00025, but this issue does not occur in model VGG08. Vanishing gradients in VGG38 makes most parameters unchanged, maintaining original random state. As depicted in Figure 1, the training and validation loss is almost non decreasing during training process, resulting much worse performance compared to VGG08 in both training and validation set. .

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) One interpretation of how the VGP arises is that stacking non-linear layers between the input and output of networks makes the connection between these variables increasingly complex. This results in the gradients becoming increasingly scrambled as they are propagated back through the network and the desired mapping between input and output being lost. He *et al.* observed this on a deep 56-layer neural network counter-intuitively achieving a higher training error than a shallower 20- layer network despite higher theoretical power. Residual networks, colloquially known as ResNets, aim to alleviate this through the incorporation of skip connections that bypass the linear transformations into the network architecture. The authors argue that this new mapping is significantly easier to optimize since if an identity mapping were optimal, the network could comfortably learn to push the residual to zero rather than attempting to fit an identity mapping via a stack of nonlinear layers. They bolster their argument by successfully training ResNets with depths exceeding 1000 layers on the CIFAR10 dataset. Prior to their work, training even a 100-layer was accepted as a great challenge within the deep learning community. The addition of skip connections solves the VGP through enabling information to flow more freely throughout the network architecture without the addition of neither extra parameters, nor computational complexity.

4. Solution overview

4.1. Batch normalization

Batch Normalization (BN) layer can be inserted between convolutional layer and activation functions to manipulate activation values to reduce generalisation errors, which has the following main effects.

1. Accelerating training process and gradient loss can be effectively mitigated, also allowing the network to support a greater learning rate with fewer learning steps to achieve the same accuracy.
2. Preventing overfitting.
3. Reducing parameter initialisation requirements and disrupting sample order.
4. Reduce the L2 weight attenuation factor.
5. Remove or use a lower dropout.

In the beginning, Batch Normalization was not proposed to solve the gradient problem, but to solve the Internal Covariate Shift problem. The calculation process for Batch Normalisation is shown in Algorithm 1.

When training, $\frac{\partial E}{\partial \gamma}$ and $\frac{\partial E}{\partial \beta}$ are calculated,γ and β are updated by gradient descent and in order to back-propagate gradient through Batch Norm layer, derivative w.r.t. them are also required: $\frac{\partial E}{\partial \mu_B}$, $\frac{\partial E}{\partial \sigma_B^2}$, $\frac{\partial E}{\partial \hat{x}_i}$, $\frac{\partial E}{\partial x_i}$.

For training process, this differentiable framework ensures layers continue learning on input distribution where less internal covariate shift occurs, thus improves training(Ioffe & Szegedy, 2015). For test time, μ_B and σ_B^2 are calculated over all mini-batch sample mean and variance, thus become a fixed transformation. Specifically,

Algorithm 1 Batch Normalisation Calculation

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$ Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

$$\mu_{test} \leftarrow E_B[\mu_B] \tag{3}$$

$$\sigma_{test}^2 \leftarrow \frac{m}{m-1} E_B(\sigma_B^2) \tag{4}$$

where the expectation is often estimated by Monte Carlo estimates: just take average of samples among batches. For processing a new test sample, simply replace μ_B and σ_B^2 in **Algorithm 1** by new estimates in (3) and (4). Vanishing gradient problem (VGP) largely related to ICS problem, changing distribution of layer inputs slows down training and make it notoriously difficult to train models with saturating nonlinearities(Ioffe & Szegedy, 2015). The BatchNorm methods, which automatically reshape input distribution for each layer by normalizing layer inputs, is able to seek a stable distribution of activation values through training. Therefore, the normalization transforms activation values back to gradient-exit regions, thus mitigates VGP.

.

4.2. Residual connections

A "residual unit", the relationship between its inputs and outputs can be expressed by the following Equations.

$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathbf{F}(\mathbf{x}_l, W_l), \tag{5}$$

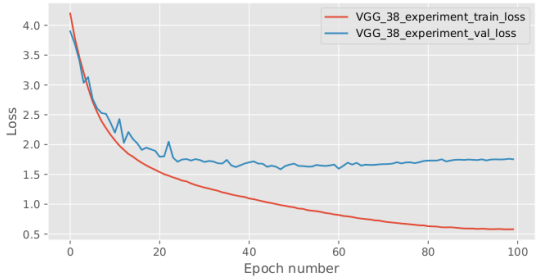
$$\mathbf{x}_{l+1} = f(\mathbf{y}_l) \tag{6}$$

where $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$ are input and output of l-th unit and l+1-th unit, W_l is referred to the parameters in layer l. $\mathbf{F}$ is the residual function(He, 2016). In (He et al., 2016), $h(\mathbf{x}_l) = \mathbf{x}_l$ and f is ReLU function.

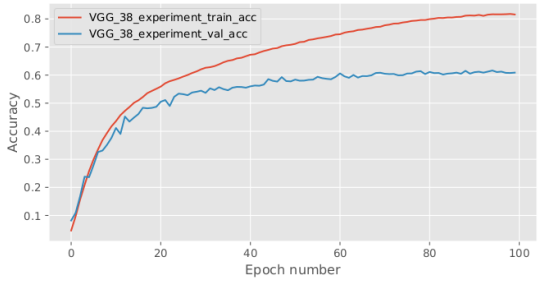
As deep learning practitioners know,since the gradient of back propagation during neural network training is very important,and if the gradient is close to 0, then the signal cannot be propagated backwards.

If assume f as an identity mapping and consider (7) and (8) recursively, then we can get

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=1}^{L-1} \mathbf{F}(\mathbf{x}_i, W_i), \tag{7}$$



(a) Loss per epoch



(b) Accuracy per epoch

Figure 4. Training curves for VGG38 BN+RC

for any deeper unit $\mathbf{L}$ and shallower unit l. Additionally, from the chain rule of back-propagation, we have:

$$\frac{\partial E}{\partial \mathbf{x}_l} = \frac{\partial E}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial E}{\partial \mathbf{x}_L}(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=1}^{L-1}\mathbf{F}(\mathbf{x}_i, W_i)). \quad (8)$$

The unique benefit of residual connection appears, the bracketed term is in form of 1 plus a value, which suggests non-zero gradient since in general $\frac{\partial}{\partial \mathbf{x}_l}\sum_{i=1}^{L-1}\mathbf{F}(\mathbf{x}_i, W_i)$ cannot always be -1,which means that the gradient from XL to Xl can be passed on all the way, and the training process of the neural network will be more efficient if the gradient can be passed on efficiently and thus alleviates VGP.

.

5. Experiment Setup

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

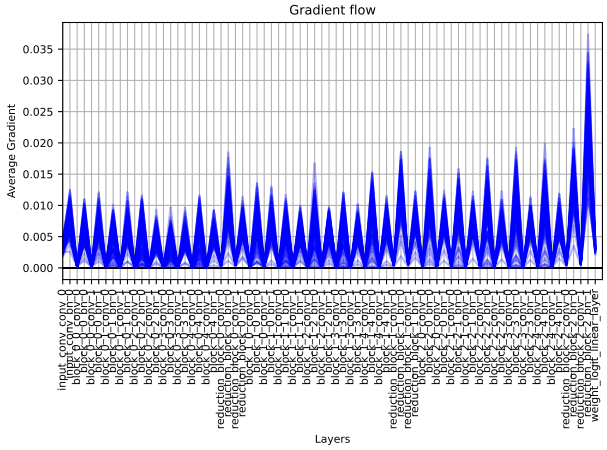


Figure 5. Gradient Flow on VGG38 BN+RC

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 4.

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)

6. Results and Discussion

Announcement: All data in Table 1 is replaced by my own experiment results in order to make valuable comparison

For Table 1,it's clear that VGG38 with BN or RC both show much better results than original VGG38, the validation(val) accuracy has dramatically improved form only 0.01% to above 46%,which means BN and RC are both effective ways to address VGP. The fact that all the modified VGG38 models achieve at least equal performance than VGG08 models agrees with the theoretical conclusion that deeper networks own greater generalization ability to a large extent.

By comparing results of 'VGG38 BN lr=1e-3' and 'VGG38 BN lr=1e-2', we observe that their val accruacies does not decrease a lot as imagined(47.07 and 43.76 respectively), same facts observed for 'VGG38 BN+RC lr=1e-3' and 'VGG38 BN+RC lr=1e-2'. This indicates that BN is more stable for large step training, allowing a larger learning rate (lr).

As for experiment results of 'VGG38 BN lr=1e-3' and 'VGG38 RC lr=1e-3', BN methods leads to 47.04% val accuracy, while RC achieves 52.24% val accuracy, higher than that of previous BN setting. We conclude that in contrast,RC is a relatively more powerful approach, as it

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	1.50	56.53	1.96	47.04
VGG38 RC	1e-3	336 K	0.90	72.63	2.18	52.24
VGG38 BN + RC	1e-3	339 K	0.65	79.58	1.75	58.24
VGG38 BN	1e-2	339 K	1.76	50.71	2.06	43.76
VGG38 BN + RC	1e-2	339 K	0.58	81.60	1.75	60.88

Table 1. Experiment results (number of model parameters, Training and Validation loss and accuracy) for different combinations of VGG08, VGG38, Batch Normalisation (BN), and Residual Connections (RC), LR is learning rate.

achieves better performance without introducing extra parameters with only 336K,which less than 339K for VGG38 with BN.

The combined experiments (BN + RC) are also conducted, and these are the ones achieving best performance among all experiments, exceeding best from others (VGG38 RC) by around 7%, which shows combining these two techniques leads to straightforward improvement. We conclude that 'VGG38 BN+RC lr=1e-2' is the best model in all of our experiments, leading to highest 81.60% train accuracy and 60.88% val accuracy. The same setting with lr=1e-3 holds quite similar performance, with val accuracy only slightly lower(58.24%). From the gradient flow plot in Figure 5, we find it indeed addresses vanishing gradient problem (VGP). When the gradient flows across BN layers, it gets enlarged before further back-propagation, thus alleviating exponentially decreasing gradient in deep network. The training curves for this setting is shown in Figure 4, differing from disability to learn parameters in plain VGG38 of figure 1, the training loss is continuously decreasing during training, suggesting the broken network has indeed been fixed.

To improve model performance, further experiment might include training with deeper VGG models. From an engineering point of view, we can use neural architecture search(NAS) to search for the optimal network architecture (VGG models) since we already have more powerful computing resources.For example, (Liu et al., 2018) proposed a progressive Neural Architecture Search method,which use reinforcement learning and evolutionary algorithms search through the structure space of convolutional neural networks (CNNs). In this way, we might be able to explore full potential of theoretically better generalization capability of deep networks.

This report only focuses on the effect of BN and RC on addressing VGP. However, there are other directions of work for understanding how these tricks success. For example, (Santurkar et al., 2018) argues that the success of batchNorm does not rely on alleviating ICS problem, however, its success is that it makes the optimization landscape significantly smoother. Thus, research on how BN and RC affect the behaviour of landscape of loss function can be listed in next step experiments. This work involves visualization a loss function. (Li, 2017) proposed a filter normalization method, which is showed to give better intuitive understanding of loss functions. Further visualization of loss landscape with and without BN and RC based on his approach can be compared to learn more about the behaviour of BN and RC.

.

7. Conclusion

The best model when evaluating on the validation set during experiments discussed in Section 4 was a model using VGG38 BN+RC with lr = 1e-2 has the best performance with val accuracy 60.88%.Our experimental results confirm the usefulness of BN and Residual Connection as a training-improve methods, but the reason of their success are remain less understood. For example,as for residual connection, people have also proposed improved versions such as adjustable shortcut connection(Li & He, 2018) and RiR structure(Targ & Lyman, 2016), both are illustrated to provide improvement based on ResNet. Thus, diving into further understanding of Batch Normalization and Residual Connection such as a multi-scale residual network fully exploits the image features is proposed in (Zhong et al., 2017), an end-toend spectral–spatial residual network focused on hyperspectral image classification tasks.In addition, the effect of Batch Normalization is argued that doesn't address ICS problem at all,(Santurkar et al., 2018) show that the fundamental reason is that it leads to a smoother loss function. While (Bjorck, 2018) argues that the faster convergence and better generalization rely on the fact that BN primarily enables larger learning rate.

Above all, i think using NAS to Search for convolutional blocks of suitable depth and size and training parameters, combined with Batch Normalization,proposing improvement based on that is a recommended future research direction.

.

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.

Bjorck, Johan, et al. Understanding batch normalization. 2018.

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, et al. Identity mappings in deep residual networks. 2016.

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.

Li, Baoqi and He, Yuyao. An improved resnet based on the adjustable shortcut connections. 2018.

Li, Hao, et al. Visualizing the loss landscape of neural nets. In *32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.*, 2017.

Liu, Chenxi, Zoph, Barret, Neumann, Maxim, Shlens, Jonathon, Hua, Wei, Li, Li-Jia, Fei-Fei, Li, Yuille, Alan, Huang, Jonathan, and Murphy, Kevin. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018.

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.

Targ, Sasha, Diogo Almeida and Lyman, Kevin. Resnet in resnet: Generalizing residual architectures. 2016.

Zhong, Zilong, Li, Jonathan, Luo, Zhiming, and Chapman, Michael. Spectral–spatial residual network for hyperspectral image classification: A 3-d deep learning framework. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2):847–858, 2017.