MLP Coursework 2: Investigating the Optimization of Convolutional Networks

s2045458

Abstract

In this report, we identified the problem behind broken 37 layer CNN in figure 1 is vanishing gradients. The reason behind the problem of vanishing gradients is too much small derivations is multiplied via chain rule in the backward propagation equations. Several potential solutions are reviewed through related literature and residual learning is selected to solve the problem of vanishing gradients. The result turned out that residual learning eliminates vanishing gradients and it has brilliant performance when training deeper neural network. A set of experiments is performed to find the best hyper-parameter value for residual network to gain the best training and generalization performance.

1. Introduction

The aim of the machine learning coursework is to explore classification of images. Different from the first machine learning coursework, this coursework uses Convolutional Neural Network (CNN), review and implements several relative technologies. An introduction to CNN (O'Shea & Nash, 2015) stated that CNN is mainly implemented to solve image-related pattern recognition tasks, which benefits from its high precision and simple architecture. Differs from other types of Artifical Neural Network that focus on the entirety problem domain, CNN exploits the solutions from the perspectives of the specific types of input, which allows setting up with simpler network architecture.

The given coursework framework provides a 7 layer and 37 layer CNN network constructed with same network layer setting. The loss curves in figure 1 and gradients flows in figure 2 reveal that the gradients reduce and finally vanish with CNN architectures varying from shallow to deep, which leads to the phenomenon that CNN is not able to minimize its loss and converge while increasing the number of layers. The problem identified from a series of zero gradients values depicted in figure 2(b) is vanishing gradients. Vanishing gradients occurs while implementing neural network with sigmoid-like activation function or training deep neural network through gradient descent methods with backward propagation (MacDonald et al., 2019). In the past few years, many researches (Bialer et al., 2019), (Xu et al., 2018), (Ma & Zhang, 2019) benefit from deep neural network. However, the problem of vanishing gradients prevent improving the performance of neural network from simply stacking layers.

In the section 3, three potential solutions to vanishing gradients are reviewed. Batch normalization addresses vanishing/exploding gradients and saturation problem (Ioffe & Szegedy, 2015). Residual learning addresses vanishing/exploding gradients and degradation problem (He et al., 2015). Deeply-supervised nets focuses on transparency of intermediate layers, robustness of obtained features and vanishing gradients (Lee et al., 2014). Considering the unique advantage on deep neural network and the accessibility on implementation, residual learning network is used to debug and fix the given broken 37 layer CNN. The kernel idea of residual learning network is creating shortcuts in the neural network. To create shortcuts, the primitive method is using identity mapping in residual building block. The details are specified in section 4. The implementation of residual network on 37 layer CNN framework shows it did solve the problem of vanishing gradients, which provides a starting point to perform experiments on how adjusting hyper-parameters could help residual network to address vanishing gradients more effectively. The information of default residual network and the corresponding hyper-parameter search experiments are stated in section 5.

To sum up, this report implements residual network to address the problem of vanishing gradients, analysis the reason of why residual network could eliminate vanishing gradients and perform experiments to find the best hyper-parameters for residual network.

2. Identifying training problems of a deep CNN

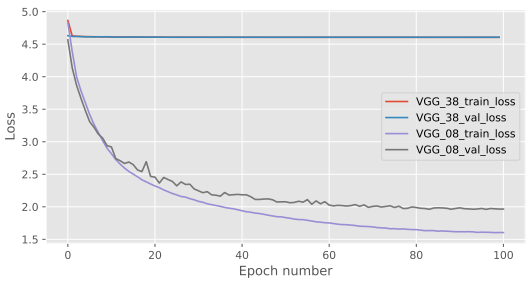
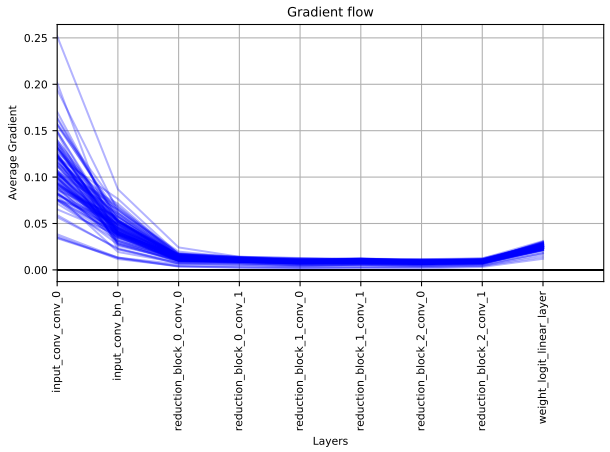
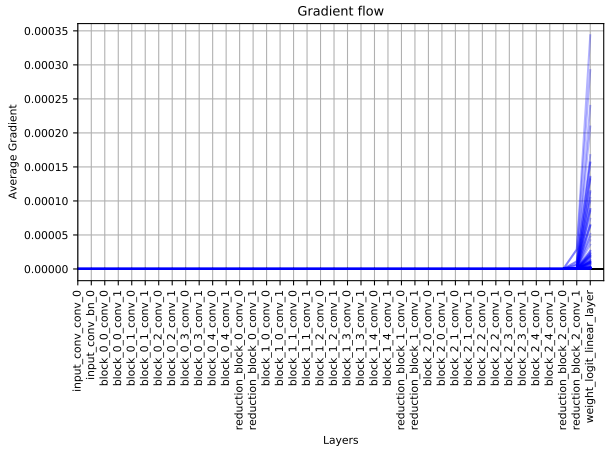


Figure 1. Training and validation plots of healthy 7 layers neural network and broken 37 layers neural network in term of loss.



(a) Healthy 7 layers neural network.



(b) Broken 37 layers neural network.

Figure 2. Gradients flows of healthy 7 layers neural network and broken 37 layers neural network.

The figure 1 shows the broken 37 layer CNN network can not continue to converge and the loss value stays constant after a few number of epochs whereas the healthy 7 layer CNN network is able to normally minimize its loss. To explore the reason why loss plots of two networks seems completely different, the gradient values of each layer in two networks are analysed. The figure 2 illustrates the gradient flow of broken 37 layer CNN network reduces to zero after a few epochs whereas the gradient flow of healthy 7 layer CNN network stabilizes during the training process, which explains the reason why the broken 37 CNN network is unable to minimize its loss. According to the figure 2(b), the problem behind the issue mentioned above is vanishing gradient. Vanishing gradients raise problems while training deep neural network through gradient descent methods with backward propagation (MacDonald et al., 2019). During the training process, a great number of derivatives are multiplied together via chain rule in the backward propagation equations. If a significant number of these derivatives are too small, the gradients will vanish, which leads to little learning in the early stage of network training (MacDonald et al., 2019).

3. Background Literature

3.1. The literature review of Batch Normalization

The paper (Ioffe & Szegedy, 2015) addressed saturation problem, which occurs while implementing sigmoid-like activation functions, such as sigmoid and tanh. After several epochs of training, the linear components of neurons will have values very big or very small, which is called vanishing gradients or exploding gradients. The normal solutions of the problem include ReLU activation function, small learning rates and appropriate initialization. The authors tried to stabilize the distributions of deep neural network internal trains, which is defined as internal covariate shift. They proposed a new mechanism called batch normalization to eliminate internal covariate shift and accelerate training. It adds a normalization step to fix the value of inputs after linear function before activation function. It calculates the mean and variance of mini-batch values and re-normalize them, which avoids the values become too larger or small. Furthermore, considering the feature of sigmoid-like activation functions, to avoid being constrained in the linear regime of the nonlinearities, batch normalization introduces a pair of parameters to scale and shift the values. These parameters are learned corresponding to the original model parameters to ensure it could recover the original activation. However, after understanding the mechanism of batch normalization, it is not difficult to find batch normalization is not appropriate for training small size batches, whose mean and variance might significantly differ from the whole training data sample. In normal cases, the paper proved the batch normalization integrated deep neural network is more tolerant to increasing learning rate and able to train with saturating nonlinearities. It also do not necessarily rely on dropout for regularization. The paper performed experiments on batch normalization network, compared to other neural networks, batch normalization ensemble model obtained lowest error rate. For vanishing gradients identified in the broken 37 layer CNN network, the batch normalization is implemented to re-normalize and fix the forward values in mini-batch to prevent values becoming too larger or small. Consequently, the fixed forward values will protect gradients from vanishing or exploding.

3.2. The literature review of Deep Residual Learning

Many recognition tasks have significantly benefited from deep conventional neural network model with a great depth. However, the problem of vanishing/exploding gradients prevents network improving by just stacking more layers. After the saturation problem is addressed by batch normalization or Relu activation function, the problem of degradation occurs. With neural network getting deeper (the number of layers increasing), accuracy gets saturated and degrades significantly after that. The reason can be explained by too many small/big derivatives are multiplied by chain rule in the backward propagation, the gradients will vanish/explode (MacDonald et al., 2019). To address degradation problem, the paper (He et al., 2015) proposed a deep residual learning framework. A residual network

is composed of a sequence of residual building blocks. In the paper, shortcut connection is embedded in the residual block and it is achieved by identity mapping. The example building block consists of two layers and the result value of the linear function in second layer is added with the input value of first layer before the activation function of second layer. A plain network and a residual network is designed by authors based on VGG network. These networks are tested under ImageNet 2012 classification dataset and CIFAR-10 dataset. The experiment on first dataset proved residual network has better performance with deeper structure whereas plain network gets worse. The experiment on second dataset indicates residual network achieves its best performance when the number of layers is around 110. The author also concluded two main advantages of residual network: easy to be optimized and to receive accuracy improvement from greatly increased network depth. For the problem of vanishing gradients identified in the broken 37 layer CNN network, the shortcut connections embedded in residual building block, which is identity mapping in this case, preserve gradients from vanishing or exploding.

3.3. The literature review of Deeply-Supervised Nets

The paper (Lee et al., 2014) focused on three classic issues of convolutional neural network type architectures: transparency of intermediate layers, robustness of obtained features, vanishing gradients. To provide a solution to these issues, the authors introduced deeply-supervised nets. The method provides supervision at each individual hidden layer with companion objective functions. The supervision from each layer is combine with supervision propagated from the output layer to update the weight in backward propagation step. Two evident advantages of deep supervision include that deep supervision function provides strong regularization for shallow network and convenience of convergence improvement for deeper network. The approach in the paper is based on an existing CNN frameworks and extended with a classifier. The model framework obtains state-of-the-art performance on four dataset: MNIST, CIFAR-10, CIFAR- 100, and SVHN. The experiments reveal that DSN could minimize or eliminate vanishing/exploding gradients to some extent. The loss function of neural network obtained regularization ability with the support of DSN. The visualised feature map in convolutional layer showed the learned features of DSN is more intuitive than normal CNN. For the vanishing gradients identified in the broken 37 layer CNN network, the companion objective function provides additional supervision at each layer. Consequently, even the gradients from backward propagation is close to zero, the gradient generated by companion objective function helps corresponding weights layer to update itself.

To sum up, all of batch normalization, residual learning and deeply-supervised nets provide feasible solutions to address vanishing gradients problem. Batch normalisation and residual learning modify the forward propagation of neural network whereas deeply-supervised nets modifies backward propagation. The performance of batch normalization is restricted by batch size, but residual learning and deeply-supervised network have no obvious shortcomings.

4. Solution Overview

In this report, the residual learning is implemented to fix the broken 37 layer CNN network. Residual network provides a feasible solution to vanishing gradient and degradation problem. Residual network demonstrates its brilliant performance in training deeper neural network compared to plain networks and the evaluation of residual network with 18 and 34 layers on ImageNet 2012 classification dataset in (He et al., 2015) proved the statement. An overview of deep learning (Schmidhuber, 2014) defined that the network with more than three layers is deep and any network with more than ten layers is very deep. In this coursework, the broken CNN network to fix has 37 layers. Such a network with a great amount of layers is ideal for residual learning to implement. In addition, the primitive structure of residual building block in (He et al., 2015) is easy to understand and implement and the given CNN coursework framework is written in the form of residual block to some extent. Both the strength and accessibility motivates the implementation of residual learning in this report.

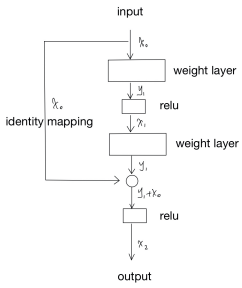


Figure 3. Primitive structure of residual building block.

Algorithm 1 ResNet block forward function pseudocode

Input: ResNet block input *in*
$x_0 = in$
$y_0 = forward(x_0)$
$x_1 = leaky\_relu(y_0)$
$x_0 = avg\_pool(x_0)$ {Only used in pooling block}
$x_1 = avg\_pool(x_1)$ {Only used in pooling block}
$y_1 = forward(x_1)$
$y_1 = y_1 + x_0$
$out = leaky\_relu(y_1)$
return *out*

The residual learning is achieved by feedforward neural network with shortcut connections (He et al., 2015). As illustrated in figure 3, a primitive residual building block consists of two layers. In this case, the shortcut connection is identity mapping, which adds the initial input of first layer to the result of second weight layer before relu activation function. The algorithm 1 specifies how the forward function works in a residual building block including both the cases of convolutional processing block and con-

volutional dimentionality reduction block in the form of pesudocode. A series of such residual building block form a residual learning network and the embedded shortcuts in the residual network allows gradients to propagate backward without decay, which solves the problem of gradients vanishing to some extent.

5. Experiments

The experiments in this section is conducted on CIFAR100, which consists of 500 training examples and 100 testing examples, in each of them contains 100 non-overlapping classes (Krizhevsky et al., 2009). In the experiments, a group of training images is used for validation. Consequently, there are 47500 image classes in training set, 2500 image classes in validation set, 10000 image classes in testing set and each of the image classes has dimension 32*32*3. The input dimension used for neural network is 32*32.

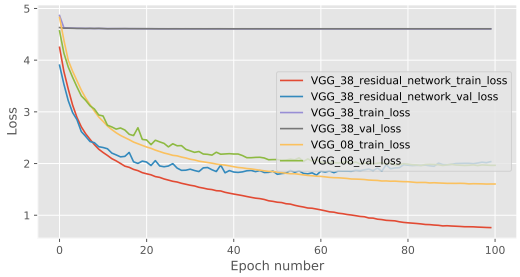


Figure 4. Training and validation plots of healthy 7 layers neural network, broken 37 layers neural network and 37 layers residual neural network in term of loss.

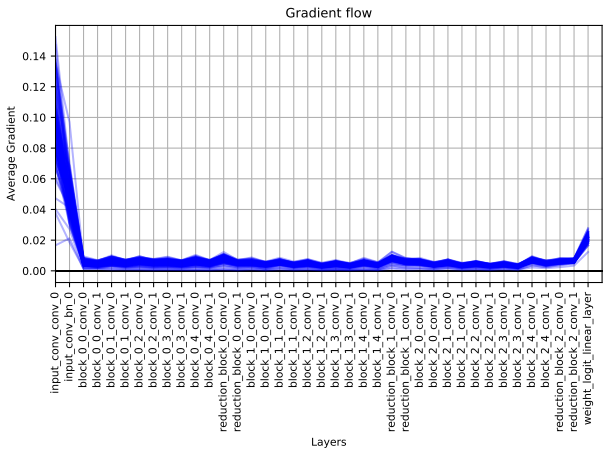


Figure 5. The gradients flow of 37 layer residual learning network.

The experiments in this section aims to find how the solution addresses the problem identified in section 2 and the best hyper-parameters that help addressing the problem and improving training and generalization. First, the primitive residual learning network is constructed with residual learning network specified in section 4 based on the coursework broken 37 layer CNN framework. Figure 4 illustrates the loss curve of primitive residual learning network compared to healthy 7 layer CNN network and broken 37 layer CNN network discussed in section 2. The phenomenon of vanishing gradients has been eliminated by implemented residual learning method and fixed 37 layer network is able to converge. Figure 5 depicts the gradients flow of residual learning network, it shows the gradients stay in a reasonable range during the training process which proves the effectiveness of residual learning method to solve vanishing gradients problem.

HYPER-PARAMETER DEFAULT SETTING	VALUE
LEARNING RATE	0.001
WEIGHT DECAY COEFFICIENT	0
BATCH SIZE	100
TRAINING ACCURACY (%)	77.2 ± 0.6
VALIDATION ACCURACY (%)	56.1 ± 0.5

Table 1. The default hyper-parameter setting and training/validation accuracy of primitive residual network.

Table 1 presents the default hyper-parameter setting of primitive residual network and the training/validation accuracy as a starting point. The training/validation loss value is obtained by setting seed values to five different values (0, 2045458, 186285, 02122020, 11132) and calculating the mean and variance. The results of hyper-parameter investigation experiments in the rest of this section is obtained by same seeds. Then, to investigate how hyper-parameters could improve the performance of residual learning network and how the adjustment of hyper-parameters could help residual learning network to address vanishing gradients more effectively. A series of valid hyper-parameter searches is performed based on the default setting. The hyper-parameters investigated in the experiments include learning rates, batch size and weight decay coefficients.

LEARNING RATE	TRAIN ACC (%)	VAL ACC (%)
0.01	1.0 ± 0	1.1 ± 0.2
0.001 (DEFAULT)	77.2 ± 0.6	56.1 ± 0.5
0.0001	50.1 ± 0.2	44.9 ± 0.2
0.00001	22.2 ± 0.4	22.1 ± 0.8

Table 2. The learning rate experiment on residual network and result validation accuracy.

To investigate the effect of each hyper-parameter, individual hyper-parameter search is performed on learning rate, weight decay coefficient and batch size separately. Table 2 indicates the default learning rate 0.001 is the most appropriate parameter setting. Excessive learning rate such as 0.01 leads to the phenomenon that network cannot converge. Too small learning rate such as 0.0001 and 0.00001 leads to underfitting. All these improper learning rates result in

WEIGHT DECAY COEFFICIENT	TRAIN ACC (%)	VAL ACC (%)
0.01	33.4 ± 1.2	32.8 ± 1.4
0.001	70.4 ± 0.8	58.9 ± 0.6
0.0001	80.3 ± 0.5	58.4 ± 0.7
0.00001	78.2 ± 0.6	57.1 ± 0.2
0 (DEFAULT)	77.2 ± 0.6	56.1 ± 0.5

Table 3. The weight decay coefficient experiment on residual network and result validation accuracy.

BATCH SIZE	TRAIN ACC (%)	VAL ACC (%)
25	66.6 ± 1.7	55.0 ± 1.0
50	73.0 ± 0.8	56.9 ± 0.9
100 (DEFAULT)	77.2 ± 0.6	56.1 ± 0.5
200	78.1 ± 0.7	55.7 ± 0.9
500	72.7 ± 0.5	53.2 ± 1.0

Table 4. The batch size experiment on residual network and result validation accuracy.

low validation accuracy. Even while using default learning rate, which has best performance in the experiment, overfitting can be observed. The overfitting can be viewed in the curves of residual learning network in figure 4. The occurrence of overfitting emphasizes the necessary to add weight decay component to the network.

Table 3 lists the experiments on weight decay coefficients from 0.01 to 0.0001 compared to default weight decay coefficient 0. Small weight decay coefficients (less than 0.0001) increase both training accuracy and validation accuracy, but the overfitting still exists. A proper weight decay coefficient 0.001 achieves the best validation accuracy in the experiment and significantly mitigate overfitting. Excessive weight decay coefficient 0.01 almost eliminates overfitting. However, it prevents network from converging and obtains a bad validation accuracy.

Table 4 lists the experiments on batch size from 25 to 500. Too large and small batch size slightly decreases validation accuracy. The batch size 50 achieves the best validation accuracy in the experiment. Compared to previous two experiments, the effect of batch size is not as significant as learning rate and weight decay coefficient.

Based on the individual hyper-parameter search, the appropriate hyper-parameter settings with good performance is combined to find the best network configuration which could achieves best training and generalization performance. For learning rate, the performance of larger and smaller learning rate is much worse compared to default learning rate. In this experiment, learning rate is fixed to 0.001. For batch size, both 50 and 100 obtained good results and the difference is not significant. Consequently, the batch size uses either 50 or 100. Similar to batch size, weight decay coefficients in this experiment ranges from 0.001 to 0.0001. Table 5 lists the hyper-parameter combi-

WEIGHT DECAY	BATCH SIZE	TRAIN ACC (%)	VAL ACC (%)
0.001	50	1.0 ± 0	0.6 ± 0
0.001	100	70.4 ± 0.8	58.9 ± 0.6
0.0005	50	73.1 ± 1.4	60.0 ± 1.0
0.0005	100	76.0 ± 0.9	59.6 ± 0.6
0.0001	50	78.5 ± 0.6	58.8 ± 1.1
0.0001	100	80.3 ± 0.5	58.4 ± 0.7

Table 5. The experiment to find the best network configuration with different hyper-parameter combinations based on the results of individual hyper-parameter search. In this experiment, learning rate is fixed to 0.001.

nations tested in the experiment. When simply combine the best weight decay coefficient 0.001 and batch size 50 parameter setting obtained in individual experiment, vanishing gradients is observed again. It is caused by the combination small batch size and large weight decay. Small batch size leads to instable convergence and large weight decay restrains weight updating. The best validation accuracy is achieved when weight decay coefficient equals to 0.0005 and batch size equals to 50. Figure 6 illustrates the performance of residual network with best configuration compared to default residual network. The training accuracy decreases and the validation accuracy increased, which means the ability to classify images is improved and the overfitting is mitigated. The result proves the training and generalization performance is improved through validation performance and overfitting status. Table 6 provides the information of best residual learning network obtained in experiments and reports its test performance.

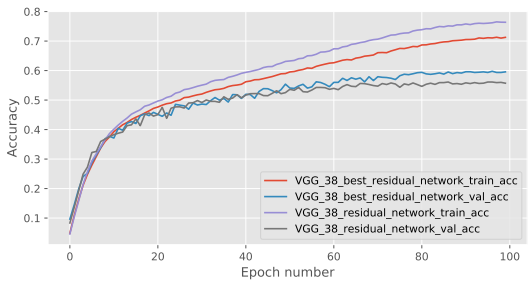


Figure 6. Training and validation plots of default residual neural network and best residual neural network in term of accuracy.

6. Discussion

From the figure 1, we can observe that the residual learning network fixed the broken 37 layer neural network and eliminates vanishing gradients. This is because shortcuts created in the residual building blocks protects the gradients from vanishing in the backward propagation. From table 2, we can observe the bad validation performance of using too large and small learning rate. This is because excessive learning rate cause the learning cannot converge

Hyper-parameter and performance	value
Learning rate	0.001
Weight decay coefficient	0.0005
Batch size	50
Training accuracy (%)	73.1 ± 1.4
Validation accuracy (%)	60.0 ± 1.0
Testing accuracy (%)	59.8 ± 1.1

Table 6. The information of best residual network obtained in the experiments.

and small learning rate cause underfitting. From table 3, we can observe appropriate weight decay coefficient could improve performance. This is because from figure 1 there exists overfitting during the training process of default residual learning network. Appropriate weight decay mitigates overfitting. From table 4, we can observe when batch size equals to 50 achieves the best validation accuracy. This is because too small batch size causes instable gradients and slows down the training efficiency. Using too large batch size is easy to stuck in local optimum. From table 6 and figure 6, we can observe the residual network with proper hyper-parameter setting has higher validation accuracy and less overfitting compared to default residual network. The proves appropriate hyper-parameter setting helps residual network gain better training and generalization performance.

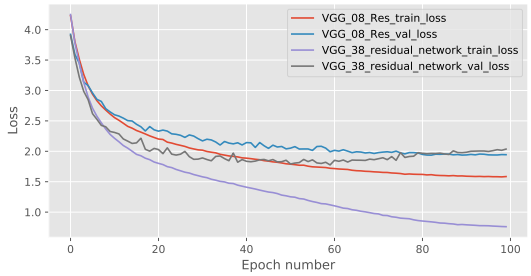


Figure 7. The comparison between 7 layer residual network and 37 layer residual network in term of loss.

This report implements residual network proposed in (He et al., 2015) to address vanishing gradients problem. the paper performed an experiment investigate the performance of 18 layer neural network and 34 layer neural network. The result shows the loss of 18 layer network is less than 34 layer network using plain network, but the loss of 18 layer network is larger than 34 layer network using residual network, which proves the strength of residual learning in deep neural network. Figure 7 compares the default 37 layer residual network with 7 layer residual network, which is constructed by implementing residual learning on 7 layer CNN network. We can conclude from figure 1 and figure 7 that residual network has special advantages in deeper neural network. The results obtained matches the conclusion in (He et al., 2015).

7. Conclusions

In this report, the optimization problems in training deep CNN architectures due to vanishing gradients problems is explored. The residual learning network is implemented to address the problem of vanishing gradients. The benefits of the residual network to this problem are demonstrated by theoretically analysing the structure of residual building blocks and embedded shortcuts and performing experiments to compare shallow and deep neural network with and without implementing residual network. The results turned out that residual network did eliminate vanishing gradients and revealed its strength in training deeper neural network. Considering only the primitive residual building block structure and identity mapping for shortcuts are implemented and discussed in the report, the work can be extended by attempting advanced residual network. For example, a multi-scale residual network fully exploits the image features is proposed in (Li et al., 2018), an end-to-end spectral–spatial residual network focused on hyperspectral image classification tasks is designed in (Zhong et al., 2018).

References

Bialer, O., Garnett, N., and Tirer, T. Performance advantages of deep neural networks for angle of arrival estimation. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3907–3911, 2019. doi: 10.1109/ICASSP.2019.8682604.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition, 2015.

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.

Lee, Chen-Yu, Xie, Saining, Gallagher, Patrick, Zhang, Zhengyou, and Tu, Zhuowen. Deeply-supervised nets, 2014.

Li, Juncheng, Fang, Faming, Mei, Kangfu, and Zhang, Guixu. Multi-scale residual network for image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Ma, Shiyong and Zhang, Zhen. Omicsmapnet: Transforming omics data to take advantage of deep convolutional neural network for discovery, 2019.

MacDonald, Gordon, Godbout, Andrew, Gillcash, Bryn, and Cairns, Stephanie. Volume-preserving neural networks: A solution to the vanishing gradient problem, 2019.

O'Shea, Keiron and Nash, Ryan. An introduction to convolutional neural networks, 2015.

Schmidhuber, Jürgen. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014. URL http://arxiv.org/abs/1404.7828.

Xu, C., Shen, J., Du, X., and Zhang, F. An intrusion detection system using a deep neural network with gated recurrent units. *IEEE Access*, 6:48697–48707, 2018. doi: 10.1109/ACCESS.2018.2867564.

Zhong, Z., Li, J., Luo, Z., and Chapman, M. 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, 2018. doi: 10.1109/TGRS.2017.2755542.