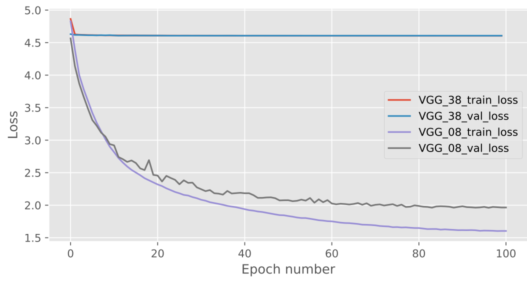
# MLP Coursework 2

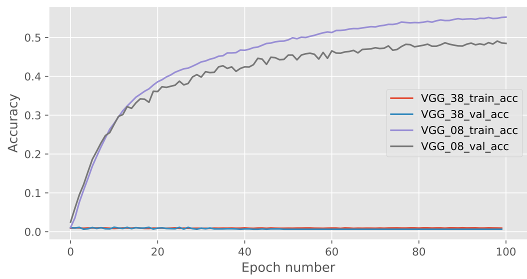
s2056595


## 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 feedforward networks (Glorot & Bengio, 2010), RNNs (Bengio et al., 1993), and CNNs (He et al., 2016). Multiple solutions have been proposed to mitigate this problem by using weight initialization strategies (Glorot & Bengio, 2010), activation functions (Glorot & Bengio, 2010), input normalization (Bishop et al., 1995), batch normalization (Ioffe & Szegedy, 2015), and shortcut connections (He et al., 2016; Huang et al., 2017).

This report focuses on diagnosing the VGP 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

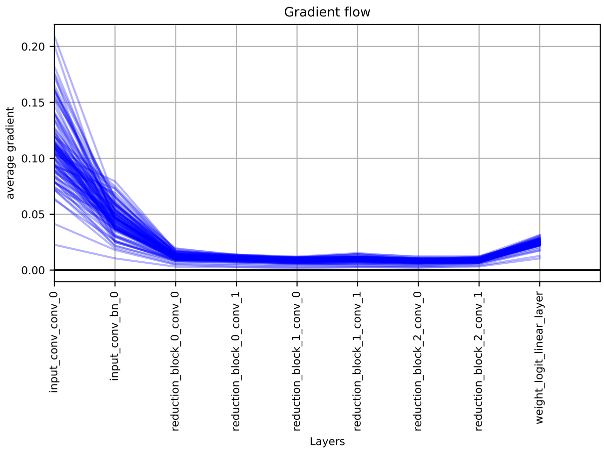



*Figure 2.* Gradient flow on VGG08

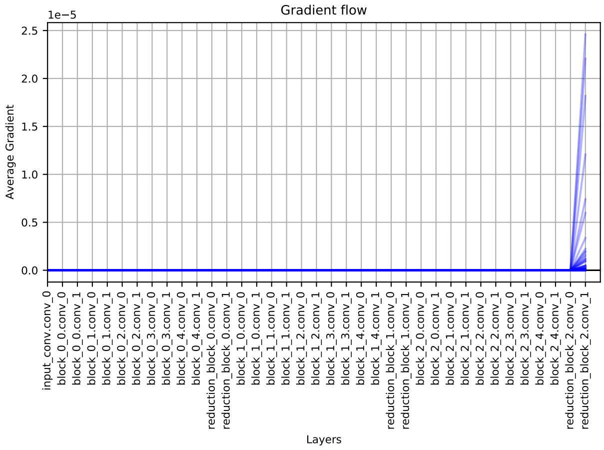



*Figure 3.* Gradient Flow on VGG38

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. 5 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. **[It can be seen from the gradient flow figure,the VGG38 model has gradient vanishing. This is because the depth network is superimposed by multi-layer nonlinear functions, and the whole depth network can be regarded as a composite nonlinear multivariate function. Then calculating the weight partial derivative of different layers for the loss function is equivalent to applying the chain law of gradient descent, which is a form of continuous multiplication. Therefore, when the number of layers is deeper, the gradient will propagate exponentially, When the number of network layers reaches tens of layers, the gradient of the number of subsequent layers will gradually tend to 0, resulting in the disappearance of the gradient. ]** .

## 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 Normal first normalizes the data, that is, subtracting the mean value and uniting the variance, as shown in equation(3). This training is faster because the data sets are mapped around the origin. However, this will lead to the poor expression ability of the later activation function, so the author introduces scaling and translation, that is, as written in equation(4).Batch Normal not only greatly improves the training speed, greatly speeds up the convergence process, but also increases the classification effect. Because it has the regularization effect of preventing over fitting similar to dropout, it can achieve considerable effect without dropout. In addition, the parameter adjustment process is much simpler, the initialization requirements are not so high, and a large learning rate can be used.**

$$\bar{x}^k = \frac{x^k - E[x^k]}{\sqrt{Var(x^k)}} \tag{3}$$

$$y^k = r^k * \bar{x}^k + \beta^k \tag{4}$$

] .

### 4.2. Residual connections

**[The main idea of residual network is to ensure that the effect of deep layer is not weaker than that of shallow layer through identity mapping.As can be seen from equation(5), the input of layer L+1 is the sum of the input and output of layer L, which ensures that when the output of layer L after passing through the convolution layer is 0, while the output of layer L+1 is not 0, because the input X of layer L is added at this time.This residual module can keep the gradient derivation in a very deep network.Experiments show that the residual network solves the degradation problem of deep neural network, and achieves very good results in image tasks such as Imagenet and cifar-10. On the premise of the same number of layers, the residual network also converges faster.**

$$F(x) = H(x, w) + x \tag{5}$$

] .

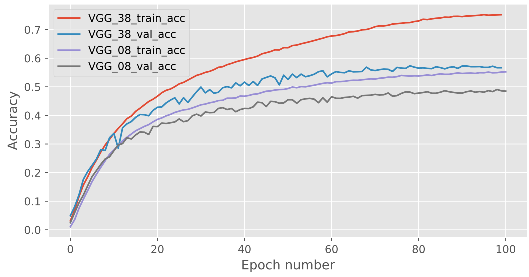



*Figure 4.* Training curves for VGG08 and VGG38

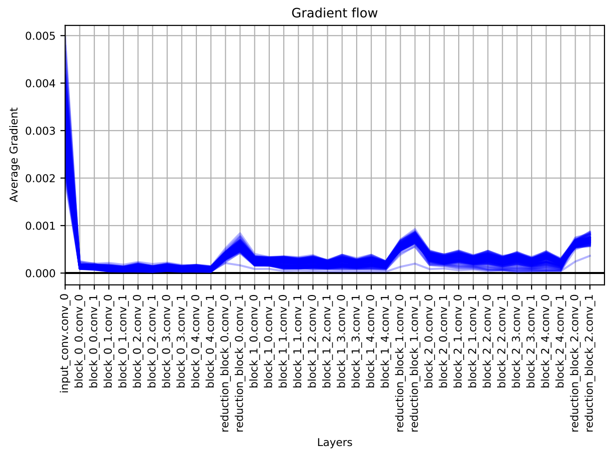



*Figure 5.* Gradient Flow on VGG38

## 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

| 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 | 1.62 | 54.09 | 2.11 | 44.96 |
| VGG38 RC | 1e-3 | 336 K | 1.33 | 61.52 | 1.84 | 52.32 |
| VGG38 BN + RC | 1e-3 | 339 K | 1.26 | 62.99 | 1.73 | 53.76 |
| VGG38 BN | 1e-2 | 339 K | 1.70 | 52.28 | 1.99 | 46.72 |
| VGG38 BN + RC | 1e-2 | 339 k | 0.66 | 79.28 | 1.74 | 60.40 |

*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.

strategies (cropping, horizontal flipping) to improve the generalization of the model.

With the goal of understanding whether BN or skip connections help fighting vanishing gradients, we first test these methods independently, before combining them in an attempt to fully exploit the depth of the VGG38 model.

All experiments are conducted using the Adam optimizer with the default learning rate (1e-3) – unless otherwise specified, cosine annealing and a batch size of 100 for 100 epochs. Additionally, training images are augmented with random cropping and horizontal flipping. Note that we do not use data augmentation at test time. These hyperparameters along with the augmentation strategy are used to produce the results shown in Figure 1.

When used, BN is applied after each convolutional layer, before the Leaky ReLU non-linearity. Similarly, the skip connections are applied from before the convolution layer to before the final activation function of the block as per Figure 2 of (He et al., 2016)

## 6. Results and Discussion

**[From the above experimental results, we can see that the network with BN layer and RC layer alleviates the disappearance of gradient and can further converge to higher accuracy. Figure 4 shows the training accuracy after improving the network. We can see that vgg38 model can converge to an accuracy close to 0.7, which is better than vgg08 model. This shows that as long as the gradient can be maintained, the model with more network layers will generally have better training effect. Figure 5 shows the gradient flow of the improved vgg38 model. It can be seen that at this time, the gradient can be well transmitted to each layer of the network. As shown in Figure 2 at the beginning, the gradient change is always 0, which also shows that BN layer and RC layer can alleviate the disappearance of the gradient Table 1 shows the quantitative research and exploration of the model by using various combinations. From the table, we can analyze the impact of each improvement method on the training results. When vgg38 does not add BN layer and RC layer, the training accuracy is close to 0 and has little classification ability. After adding BN layer, the training accuracy reaches about 0.5, and after adding BN and RC layer, the training accuracy reaches 0.7. Table 1 intuitively shows the improvement effect of BN layer and RC module on the network.]** .

## 7. Conclusion

**[In the training process, with the deepening of the network, the distribution gradually changes, resulting in the overall distribution gradually moving to the saturation interval of the activation function, so that the gradient at the bottom disappears during back propagation, that is, the reason for the slower and slower convergence. Through this experiment, we intuitively show the gradient change during the training of a 30 layer network, and optimize it with batch normal and residual module to alleviate the disappearance of the gradient. When the number of layers of neural network continues to deepen, it will be more and more difficult to optimize. At this time, we should strengthen the propagation of features between network layers, encourage the reuse of features, and reduce parameters, such as DenseNet, which is a main direction of deep network optimization. The network of layer N can be connected with the network of layer n-1. Dense connection has many advantages, which increases the transfer of gradient, reuses features, and even reduces the over fitting on small sample data.]** .

## References

Bengio, Yoshua, Frasconi, Paolo, and Simard, Patrice. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pp. 1183–1188. IEEE, 1993.

Bishop, Christopher M et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

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

*Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, Gao, Liu, Zhuang, Van Der Maaten, Laurens, and Weinberger, Kilian Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

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

LeCun, Yann A, Bottou, Léon, Orr, Genevieve B, and Müller, Klaus-Robert. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, and Mądry, Aleksander. How does batch normalization help optimization? In *Proceedings of the 32nd international conference on neural information processing systems*, pp. 2488–2498, 2018.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.