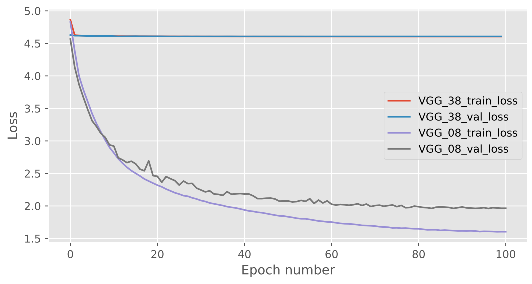
MLP Coursework 2

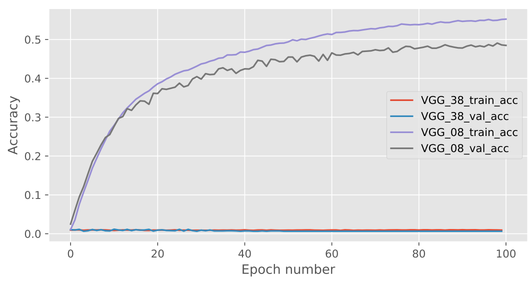
s1817967

Abstract

Deep neural networks have become the state-of-the-art in many standard computer vision problems thanks to their powerful representations and availability of large labeled datasets. While very deep networks allow for learning more levels of abstractions in their layers from the data, training these models successfully is a challenging task due to problematic gradient flow through the layers, known as vanishing/exploding gradient problem. 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 Vanishing Gradient Problem (VGP), a phenomenon where the gradients of the error function with respect to network weights shrink to zero, as they backpropagate to earlier layers, hence preventing effective weight updates. This phenomenon is prevalent and has been extensively studied in various deep neural networks 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 occurring in the VGG38 model and addressing it by implementing two standard solutions. In particular, we first study a "broken" network in terms of its gradient flow, norm of gradients with respect to its 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 mitigate the vanishing/exploding gradient problem, therefore enabling effective training of the VGG38 model, the best results are obtained by combining both BN and RC.

2. Identifying training problems of a deep CNN

[Question Figure 3 - Done]

Concretely, training deep neural networks 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 $\boldsymbol{x}^0$ to the network and producing the network prediction and also the error

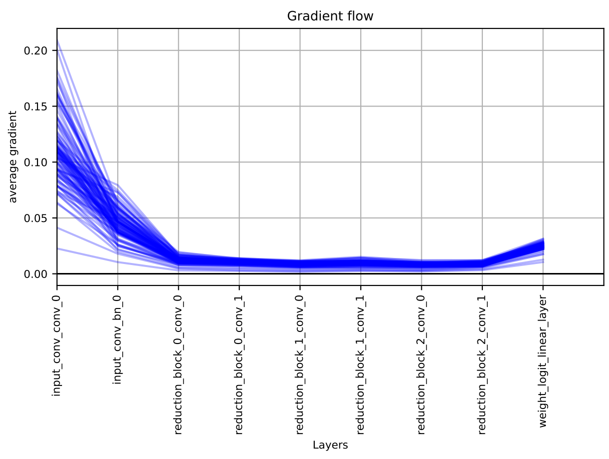


Figure 2. Gradient flow on VGG08

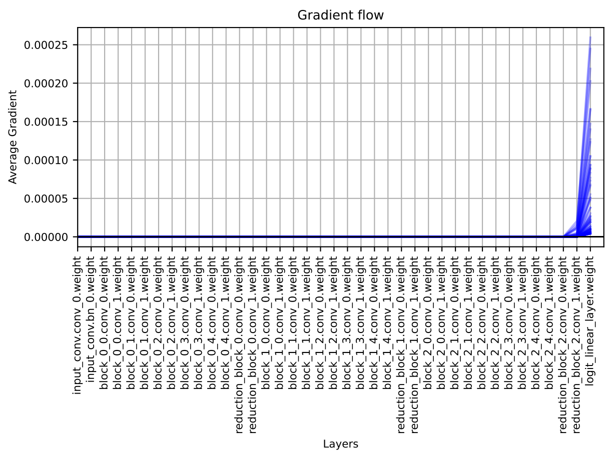


Figure 3. Gradient Flow on VGG38

value. In detail, each layer takes in the output of the previous layer 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:

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

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. **[When comparing the training curves for VGG08 and VGG38 in Figure 1, we see that, when assessed on both the training and validation sets, the loss and accuracy for VGG38 remain constant or fail to converge to a desirable value after the initial epoch, whereas the loss and accuracy for VGG08 decreases and increases, respectively, with epoch number. This indicates that learning either doesn't happen or happens very slowly for the VGG38 model, and the reason for this can be found by contrasting the gradient flows between the two models. As we backpropagate from the last layer to the first layer for the VGG08 model, it is evident from Figure 2 that the mean absolute gradient values for the weights do not reach 0. Hence, we successfully update the weights during training time to minimise the loss because gradients are never zero for each layer. In contrast, Figure 3 shows the gradient flow for the VGG38 model, where all layers except the final two have mean absolute gradient values of 0. The gradients for the preceding layers become zero after backpropagating for two layers, possibly due to the use of incredibly small partial gradient terms in the chain rule as shown in equation 2. As a result, the weights in these initial layers stay the same throughout training, which keeps the value of the loss function constant. We can infer that the model VGG38 exhibits the previously mentioned VGP.]** .

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) A well-known way of mitigating the VGP is proposed by He *et al.* in (He et al., 2016). In their paper, the authors depict the error curves of a 20 layer and a 56 layer network to motivate their method. Both training and testing error of the 56 layer network are significantly higher than of the shallower one.

[The range of mapping functions that a machine learning model can learn depends on its capacity. A model that has too much capacity may memorise the training dataset, which will cause it to overfit, whereas a model that has too little capacity will be unable to learn the training dataset and will underfit. However, overfitting is surprisingly not the cause of the higher training error by the 56 layer network. Instead, the authors of (He et al., 2016) suggest that adding more layers to a suitably deep model leads to a higher training error. Such problem is known as the vanishing gradient problem (VGP). As is well known, when backpropagation is used in neural networks, the loss is sent backward and tends to get less with each additional layer. The loss has already decreased so much by the time it is propagated back toward the initial few layers that the weights hardly change at all. It is impossible to modify and train the weights of the first few layers when such a small loss is propagated backward, and this is VGP.

Another problem that might be the cause of these results is the degradation problem. As the number of layers is increased, the training accuracy starts saturating and then degrading. This is because deeper model is harder to train and optimize. For example, consider the deeper model is constructed by adding identity layers to the shallower model. One would expect the deeper model to perform at least as good as the shallower model. But the truth is that the deeper model would have a worse performance due to the degradation problem.] .

Residual networks, colloquially known as ResNets, aim to alleviate VGP 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

[The VGP can be reduced via batch normalisation (BN). We make sure that $|x_i|$ no longer responds strongly to small changes in parameters and is within an appropriate range. This ensure that activation functions like logistic sigmoid or tanh do not saturate, preventing extremely small gradients that slow convergence by normalising, shifting, and scaling the inputs x_i to each layer or activation. The following equations can be used to summarise the steps involved in implementing BN.

$$\hat{u}_i = \frac{u_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{3}$$

$$z_i = \gamma_i \hat{u}_i + \beta_i = batchNorm(u_i) \tag{4}$$

where u_i is a hidden unit activation, $w^T x_i$ and ϵ is a small value added to variance to avoid division by zero. Sample mean μ_i and variance σ^2 can be calculated as follows.

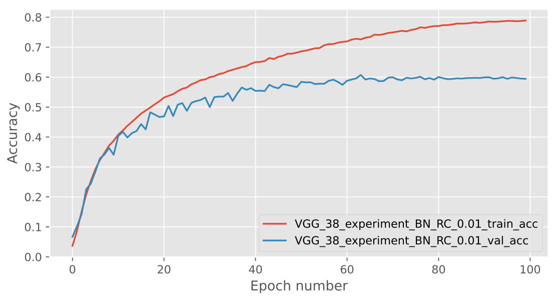
$$\mu_i = \frac{1}{M} \sum_{m=1}^{M} u_i^{(m)} \tag{5}$$

$$\sigma^2 = \frac{1}{M} \sum_{m=1}^{M} (u_i^{(m)} - \mu_i) \tag{6}$$

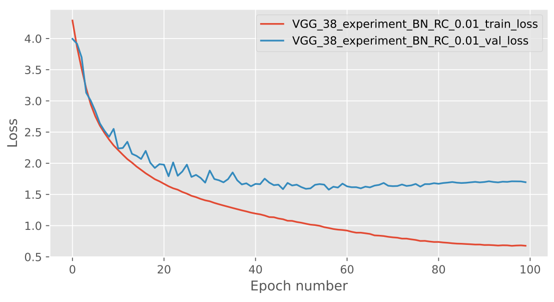
The standardised hidden units $\hat{u}_i$ are then subjected to the bathNorm() function as shown in equation 4, which essentially scales and shifts them using parameters γ_i and β_i that can be learned during training. This can ensure that they have new means and standard deviations in order to keep the standardised activations from having values that are too small for the activation functions and maintain the expressive power of the network. The procedures mentioned can be used to implement batch normalisation while training is taking place. The same holds true during testing, with the exception of the sample mean and variance used because we only have a single sample and no minibatches. In this instance, the mean and variance over the entire training set are used to calculate the mean and variance for each layer.] .

4.2. Residual connections

Residual connections are another approach used in the state-of-the-art Residual Networks (He et al., 2016) to tackle the



(a) Accuracy curves



(b) Loss curves

Figure 4. Training curves for VGG38 BN + RC with LR of 1e-2

vanishing gradient problem. Introduced by He et. al. (He et al., 2016), a residual block consists of a convolution (or group of convolutions) layer, "short-circuited" with an identity mapping. More precisely, given a mapping $F^{(b)}$ that denotes the transformation of the block b (multiple consecutive layers), $F^{(b)}$ is applied to its input feature map $\boldsymbol{x}^{(b-1)}$ as $\boldsymbol{x}^{(b)} = \boldsymbol{x}^{(b-1)} + F(\boldsymbol{x}^{(b-1)})$.

Intuitively, stacking residual blocks creates an architecture where inputs of each blocks are given two paths : passing through the convolution or skipping to the next layer. A residual network can therefore be seen as an ensemble model averaging every sub-network created by choosing one of the two paths. The skip connections allow gradients to flow easily into early layers, since

$$\frac{\partial \boldsymbol{x}_{(b)}}{\partial \boldsymbol{x}^{(b-1)}} = \mathbb{1} + \frac{\partial F(\boldsymbol{x}^{(b-1)})}{\partial \boldsymbol{x}^{(b-1)}} \tag{7}$$

where $\boldsymbol{x}^{(b-1)} \in \mathbb{R}^{H \times W \times C}$ and $\mathbb{1}$ is a $\mathbb{R}^{H \times W \times C}$-dimensional tensor with entries 1. Importantly, $\mathbb{1}$ prevents the zero gradient flow.

5. Experiment Setup

[Question Figure 4 - Done]

[Question Figure 5 - Done]

[Question Table 1 - Done]

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.

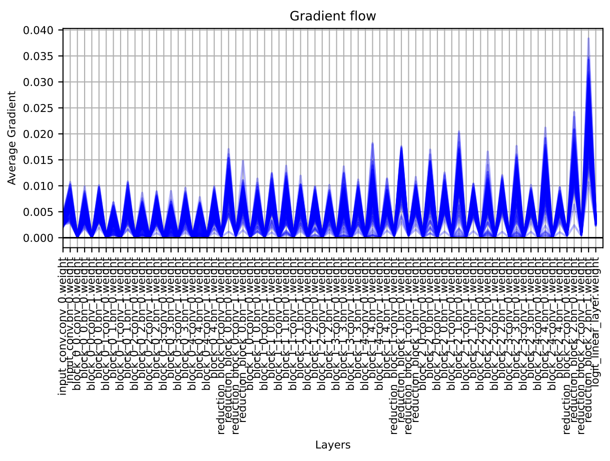


Figure 5. Gradient Flow on VGG38 BN + RC with LR of 1e-2

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). Note that adding residual connections between the feature maps before and after downsampling requires special treatment, as there is a dimension mismatch between them. Therefore in the coursework, we do not use residual connections in the down-sampling blocks. However, please note that batch normalization should still be implemented for these blocks.

5.1. Residual Connections to Downsampling Layers

[In our experiments, residual connections are not added to the downsampling layers. In order to incorporate residual connections to the downsampling layers, the experiment should be changed such that strided convolutions or a pooling layer is used. When using a pooling layer, a two-dimensional filter is slid across each channel of the feature map during the pooling procedure,

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	1.61	54.75	1.99	46.88
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	0.79	67.74	1.70	59.44

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.

and the features contained within the region covered by the filter are summarised. The benefits of using pooling layer are reducing the number of parameters of the model thus reducing the computation cost of the model. The pooling layer summarises the features in a region of the feature map produced by a convolution layer. As a result, subsequent operations are carried out on summarised features rather than precisely positioned features generated by the convolution layer. Therefore, the model is more resistant to changes in the position of the features in the input image. On the other hand, stride can compresses multiple smaller convolutions into one bigger convolution. For example, the first layer of ResNet is a strided convolutions. It compresses multiple 3x3 convolutions into one 7x7 convolution, significantly reducing the computation cost. Despite of this benefit, pooling is a fixed operation and replacing pooling with strided convolution can increase the number of parameters of the model. This can be viewed as an advantage and disadvantage at the same time. Because increasing the model's parameter can increase the model's ability to learn but also increase the computation cost of the model.] .

6. Results and Discussion

[According to Table 1, the base deeper network VGG38 appears to have no predictive potential at all given its accuracy of 0.01%, whereas the shallower network VGG08 performs noticeably better. This has been explained in prior discussions as being brought on by the VGP that deeper networks frequently experience. In order to enhance the performance of VGG38, following experiments incorporating BN and RC are carried out.

We start by examining the outcomes from using BN with a $1e-3$ initial default learning rate (VGG38 BN $1e-3$). As predicted, the model's performance has greatly increased, even matching the base VGG08 model's outcomes while having a greater training accuracy but a slightly worse validation accuracy (a difference of 0.04%). This suggests that BN does able to solve the vanishing gradient problem. In light of the fact that BN lessens the dependency of gradients on the scale of the parameters, it is possible to employ a higher learning rate (Ioffe & Szegedy, 2015). As a result, we conducted additional testing by using BN with a initial learning rate of $1e-2$. Although the training accuracy when using $1e-3$ as the learning rate is higher (54.75%) than that when using $1e-2$ of learning rate (52.28%), it appears that there is no gain in the validation accuracy as both the VGG38 BN models which employ various initial learning rates have validation accuracy of 46.88& for the model with smaller learning rate and 46.72% for the model with bigger learning rate.

Next, we examine the outcomes when RC is used with a $1e-3$ initial default learning rate (VGG38 RC $1e-3$). In comparison to when we use BN, it seems that the performance in terms of loss and accuracy are all better. Its validation accuracy is 5.44% better than the model employing BN's. This would suggest that RC is more effective at addressing VGP in deep networks. The generalisation gap (difference between val loss and train loss) for VGG38 BN $1e-3$ is smaller than that of VGG38 RC model (0.38 against 0.51). This might be due to the BN's additional benefit of acting as a regularizer, which eliminates the requirement for additional regularisation methods (Ioffe & Szegedy, 2015). Again, more tests should have been performed to accurately confirm this feature. It is possible to test experiments that contrast models that use BN alone with models that use BN and other regularisation strategies like dropout or weight penalties. As was previously indicated, RC is currently implemented in blocks without dimensionality reduction, maintaining the same dimensions for both inputs and outputs and relying solely on the identity mapping of x. We may think of incorporating RC in dimensionality reduction blocks as an effort to increase performance since (He et al., 2016) have shown that projection shortcuts lead to somewhat better performance when more parameters are added.

Finally, we tested models that combined BN and RC, and we found that they performed better than any of the earlier models that employed BN and RC alone. The model with the highest overall performance was the one with the bigger initial learning rate of $1e-2$ (VGG38 BN+RC $1e-2$), whose training curves and gradient flow plots are displayed in Figures 4 and 5. We can clearly see that the average gradients for VGG38 BN+RC $1e-2$ as they backpropagate are non zero when comparing

its gradient flow plot to the one obtained for VGG38 in Figure 2, demonstrating that VGP has been adequately addressed. The generalisation gap for VGG38 BN+RC $1e-2$ is very significant (0.91) compared to the other models, unlike when we merely used BN, where modifying the initial learning rate has a small effect on the outcomes. This may suggest that, despite being the best-performing model, the training set is causing the model to overfit. Because of this, there is still a chance that it will provide a greater generalisation and hence a better outcomes by employing additional regularisation approaches.

When comparing the performance of the VGG38 BN and VGG38 BN+RC models, we find that the latter model performs significantly better. It is suggested that such an improvement is not attributable to resolving the VGP because it has already been mitigated by BN based on the experimental setting in (He et al., 2016). Instead, by including RC in the VGG38 BN, we may have addressed the degradation problem, which is a problem in deep networks where adding more depth causes a network to perform worse on both test and training data. This may help to explain why our performance has increased when BN and RC are combined. The best performing model, VGG38 BN+RC 1e-2, was further assessed. On the test set, it produced results that were not significantly different from those on the validation set, with a loss of about 1.69 and an accuracy of 58.78%.] .

7. Conclusion

[The vanishing gradient problem (VGP), which frequently occurs during the training of deep convolutional neural networks, makes it difficult to successfully learn the weights. So as to address the VGP, we suggested techniques like Batch Normalization (BN) and Residual Connections (RC). Through experiments, we demonstrated that BN and RC are effective in addressing the VGP by contrasting VGG38 models that incorporated these methods against the original VGG38 baseline. In fact, combining the two methods yields the best results, with a test accuracy of 58.78%.

However, we do think that this may be further enhanced by investigating alternative approaches. For instance, (Huang et al., 2017) suggested adopting Dense Convolution Network (DenseNet), another type of skip connection. Each layer in DenseNet acquires weights from all of its preceding layers by concatenating the output feature maps of the preceding layer and the next layer. In order to handle both the VGP and degradation problem, shorter routes are built that maximise information flow since characteristics retrieved from early levels are directly utilised by later layers. In addition, compared to our present techniques, these networks may be trained with better computational and memory efficiency and with fewer parameters.] .

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.