MLP Coursework 2

s2713719

Abstract

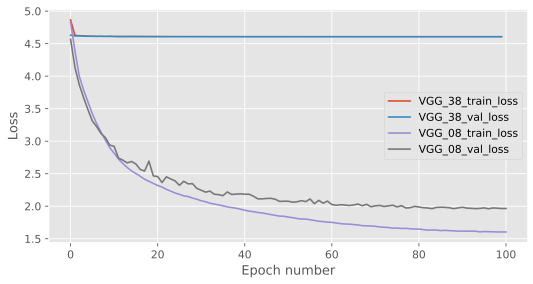
Deep neural networks have become the state-of-the-art in many standard computer vision problems thanks to their powerful representations and availability of large labeled datasets. While very deep networks allow for learning more levels of abstractions in their layers from the data, training these models 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.

1. Introduction

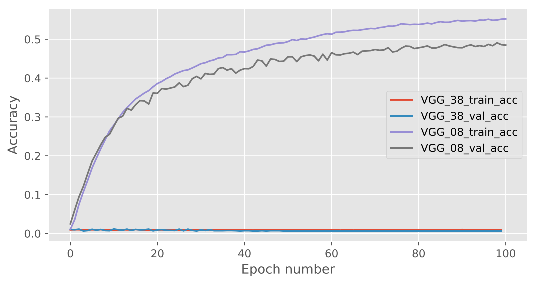
Despite the remarkable progress of modern convolutional neural networks (CNNs) in image classification problems (Simonyan & Zisserman, 2014; He et al., 2016a), training very deep networks is a challenging procedure. One of the major problems is the Vanishing Gradient Problem (VGP), a phenomenon where 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., 2016a). 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., 2016a; Huang et al., 2017).

This report focuses on diagnosing the VGP occurring in the VGG38 model[1] and addressing it by implementing two standard solutions. In particular, we first study a "broken" network in terms of its gradient flow, L1 norm of gradients with respect to its weights for each layer and contrast it to ones in the healthy and shallower 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., 2016a) 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 CIFAR100 (pronounced as 'see far 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.

[1]VGG stands for the Visual Geometry Group in the University of Oxford.



(a) Cross entropy error per epoch



(b) Classification accuracy per epoch

Figure 1. Training curves for VGG08 and VGG38 in terms of (a) cross-entropy error and (b) classification accuracy

2. Identifying training problems of a deep CNN

[Question Figure 3 - Replace this image with a figure depicting the average gradient across layers, for the

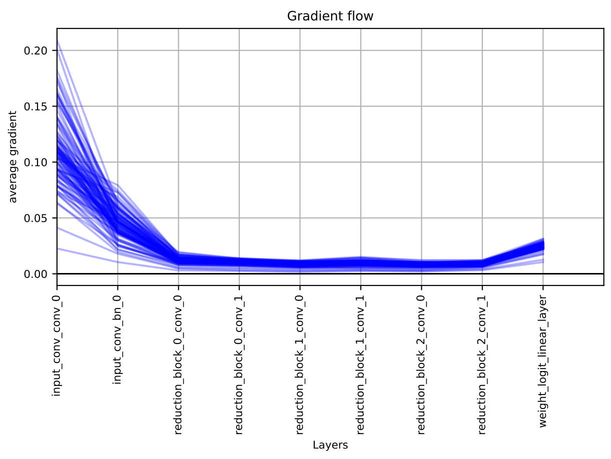


Figure 2. Gradient flow on VGG08

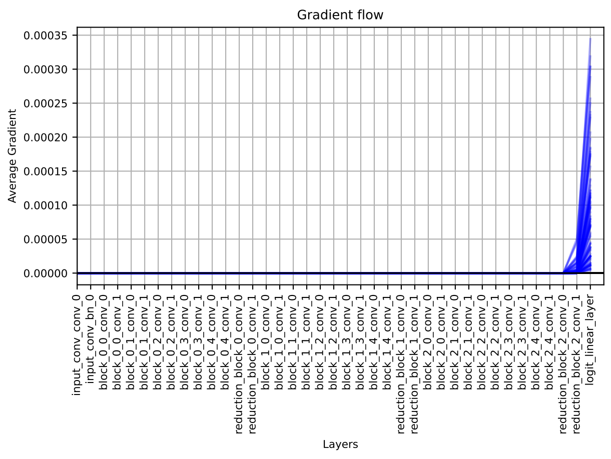


Figure 3. Gradient Flow on VGG38

VGG38 model.

(The provided figure is correct, and can be used in your analysis. It is partially obscured so you can get credit for producing your own copy).]

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 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:

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

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. **[Question 1 - As observed in Figure 3, VGG38 suffers from the vanishing gradient problem, where gradients diminish to near zero in earlier layers, hindering the model from learning and performing effective weight updates. In contrast, VGG08, as shown in Figure 2, maintains consistent gradient flow, where gradients remain above zero across layers, enabling effective weight updates and the capture of meaningful patterns. This difference is reflected in Figures 1a and 1b, where VGG38 shows high losses (4.61) and near-zero accuracy (0.01%) on both training and validation sets, which extends to poor test performance, approaching the same values. However, VGG08 achieves better results with training losses of around 1.74, validation losses of around 1.95, and accuracy near 50%. These results highlight the importance of consistent gradient flow for stable learning, efficient weight updates, and better generalization.**

] .

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., 2016a) A well-known way of mitigating the VGP is proposed by He *et al.* in (He et al., 2016a). 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.

[Question 2 - The results in Figure 1 demonstrate the degradation problem, where increasing network depth leads to higher training error. This arises from optimization challenges rather than overfitting, as the deeper network fails to fit the training data effectively. It is also not caused by vanishing gradients, which have been mitigated by techniques such as normalized initialization and intermediate normalization. Deeper networks should perform as well as shallower ones if additional layers operated as identity mappings. However, according to (He et al., 2016a), it is difficult for deeper networks to approximate these mappings effectively, making optimization increasingly challenging and emphasizing the need for improved methods to address these issues.] .

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

BN has been a standard component in the state-of-the-art convolutional neural networks (He et al., 2016a; Huang et al., 2017). Concretely, BN is a layer transformation that is performed to whiten the activations originating from each layer. As computing full dataset statistics at each training iteration would be computationally expensive, BN computes batch statistics to approximate them. Given a minibatch of B training samples and their feature maps $X = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^B)$ at an arbitrary layer where $X \in \mathbb{R}^{B\times H\times W\times C}$, H, W are the height, width of the feature map and C is the number of channels, the batch normalization first computes the following statistics:

$$\mu_c = \frac{1}{BWH}\sum_{n=1}^{B}\sum_{i,j=1}^{H,W} \boldsymbol{x}^n_{cij} \tag{3}$$

$$\sigma^2_c = \frac{1}{BWH}\sum_{n=1}^{B}\sum_{i,j=1}^{H,W} (\boldsymbol{x}^n_{cij} - \mu_c)^2 \tag{4}$$

where c, i, j denote the index values for y, x and channel coordinates of feature maps, and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the mean and variance of the batch.

BN applies the following operation on each feature map in batch B for every c, i, j:

$$\text{BN}(\boldsymbol{x}_{cij}) = \frac{\boldsymbol{x}_{cij} - \mu_c}{\sqrt{\sigma^2_c + \epsilon}} * \gamma_c + \beta_c \tag{5}$$

where $\gamma \in \mathbb{R}^C$ and $\beta \in \mathbb{R}^C$ are learnable parameters and ϵ is a small constant introduced to ensure numerical stability.

At inference time, using batch statistics is a poor choice as it introduces noise in the evaluation and might not even be well defined. Therefore, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are replaced by running averages of the mean and variance computed during training, which is a better approximation of the full dataset statistics.

Recent work has shown that BatchNorm has a more fundamental benefit of smoothing the optimization landscape during training (Santurkar et al., 2018) thus enhancing the predictive power of gradients as our guide to the global minimum. Furthermore, a smoother optimization landscape should additionally enable the use of a wider range of learning rates and initialization schemes which is congruent with the findings of Ioffe and Szegedy in the original BatchNorm paper (Ioffe & Szegedy, 2015).

4.2. Residual connections

Residual connections are another approach used in the state-of-the-art Residual Networks (He et al., 2016a) to tackle the vanishing gradient problem. Introduced by He et. al. (He et al., 2016a), a residual block consists of a convolution (or group of convolutions) layer, "short-circuited" with an identity mapping. More precisely, given a mapping $F^{(b)}$

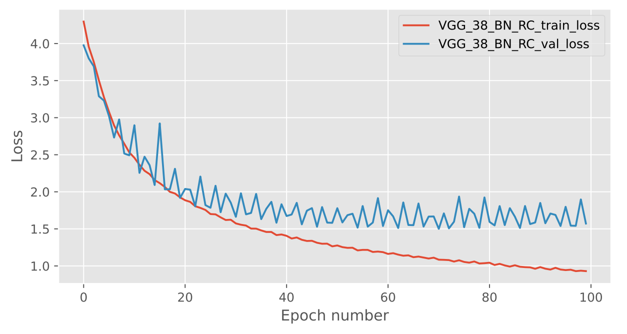


Figure 4. Training curves for VGG38 in terms of (a) cross-entropy error, with Batch Normalization (BN), Residual Connections (RC), and a learning rate (LR) of 1e-2

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

$$\frac{\partial \boldsymbol{x}^{(b)}}{\partial \boldsymbol{x}^{(b-1)}} = \mathbb{1} + \frac{\partial F(\boldsymbol{x}^{(b-1)})}{\partial \boldsymbol{x}^{(b-1)}} \tag{6}$$

where $\boldsymbol{x}^{(b-1)} \in \mathbb{R}^{C \times H \times W}$ and $\mathbb{1}$ is a $\mathbb{R}^{C \times H \times W}$-dimensional tensor with entries 1 where C, H and W denote the number of feature maps, its height and width respectively. Importantly, $\mathbb{1}$ prevents the zero gradient flow.

5. Experiment Setup

[Question Figure 4 - Replace this image with a figure depicting the training curves for the model with the best performance *across experiments you have available (you don't need to run the experiments for the models we already give you results for)*. Edit the caption so that it clearly identifies the model and what is depicted.

]

[Question Figure 5 - Replace this image with a figure depicting the average gradient across layers, for the model with the best performance *across experiments you have available (you don't need to run the experiments for the models we already give you results for)*. Edit the caption so that it clearly identifies the model and what is depicted.]

[Question Table 1 - Fill in Table 1 with the results from your experiments on

1. ***VGG38 BN (LR 1e-3)*, and**
2. ***VGG38 BN + RC (LR 1e-2)*.**

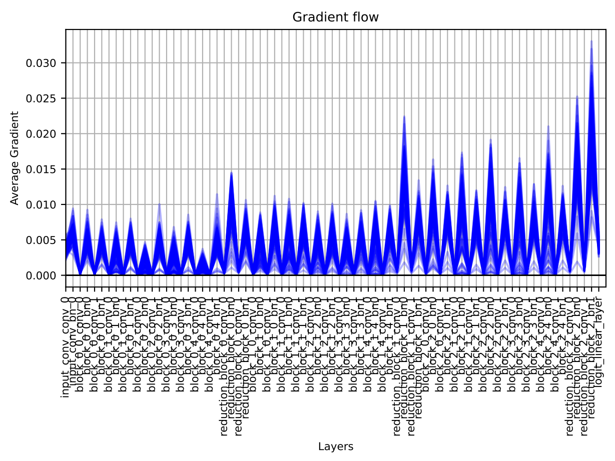


Figure 5. Gradient Flow for VGG38 with Batch Normalization (BN), Residual Connections (RC), and a learning rate (LR) of 1e-2.

]

We conduct our experiment on the CIFAR100 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 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 Fig. 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 Fig. 2 of (He et al., 2016a). 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

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	1.67	52.91	1.89	48.92
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	0.95	71.50	1.52	60.08

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.

[Question 3 -

In the VGG38 model, residual connections are excluded from the downsampling layers due to the dimensional mismatch caused by pooling operations, which reduce the spatial dimensions of feature maps. This prevents direct element-wise addition between the input x and the transformed output $F(x)$. According to (He et al., 2016a), two methods are used to resolve this mismatch and incorporate the residual connections into the downsampling layers: projection shortcuts with 1x1 convolutions and identity mapping with zero padding.

The first method, projection shortcuts with 1x1 convolutions, involves applying a 1x1 convolution to the input x to adjust its dimensions to match the downsampled output $F(x)$. This ensures that the input and output can be added together for the residual connection. As mentioned in (He et al., 2016b), this method is flexible and can effectively handle both spatial and channel mismatches, preserving the benefits of residual learning, such as improved gradient flow and feature reuse. However, it introduces extra computation because of the additional convolutional layer, increasing both the number of parameters and the computational cost.

Alternatively, identity mapping with zero padding involves adding zeros to the input x to match the dimensions of the output $F(x)$ size. This method is more computationally efficient than 1x1 convolutions, as it avoids the need for additional convolutional layers. However, it has limitations. Zero padding is less effective when significant dimensional mismatches occur because the padded regions do not contribute to the learning process. This can hinder the model's ability to capture important features, particularly in deeper networks, as highlighted by (He et al., 2016b)

Both methods require modifications to the forward pass. For 1x1 convolutions, the input x is processed through an additional convolutional layer to adjust its dimensions. In contrast, for zero padding, the input is padded to match the dimensions of the output before the residual addition.

Considering the advantages and limitations of each method, 1x1 convolutional projection shortcuts are the better choice. They offer more flexibility and preserve the benefits of residual learning. Despite the additional computational cost, this overhead does not significantly affect performance in our case. On the other hand, while zero padding is simpler and computationally lighter, it is less effective when dealing with large dimensional mismatches and deeper networks.

] .

6. Results and Discussion

[Question 4 -

From Table 1, the shallow VGG08 achieves 46.84% validation accuracy and a loss of 1.95, while the deeper VGG38 baseline fails to optimize effectively, with 0.01% accuracy and a validation loss of 4.61. This demonstrates the vanishing gradient problem in deeper networks, where optimization becomes increasingly difficult as depth increases, aligning with the challenges outlined by (He et al., 2016a).

Adding BN to VGG38 helps address the vanishing gradient issue by normalizing layer activations and reducing internal covariate shift (ICS). This allows the network to learn more effectively, improving validation accuracy to 48.92% and validation loss to 1.89. However, despite these improvements, the performance remains comparable to the shallower VGG08, indicating that BN only partially mitigates the optimization challenges deeper networks face.

On the other hand, Residual Connections (RC) significantly improve VGG38's performance by addressing optimization challenges in deeper networks. As networks get deeper, they become harder to train because the layers struggle to learn effectively. RC contribute by allowing the network to learn residual mappings, $F(x) = H(x) - x$, instead of learning full transformations. This makes optimization easier and helps the network learn more efficiently, leading to better performance. As noted by (He et al., 2016a), RC enable deeper networks to train more effectively, as shown by the improvement in validation accuracy of 52.32%, and the validation loss of 1.84.

Observing Figures 4 and 5, the combination of BN and RC with a learning rate of 0.01 is the best-performing

model, achieving a validation accuracy of 60.08% and a validation loss of 1.52, with its effectiveness reflected in test performance at 59.89% accuracy and a test loss of 1.54. This happens because the learning rate accelerates convergence, Batch Normalization stabilizes activations, and Residual Connections address the degradation problem, simplifying optimization. Together, these methods offer the most effective solution to the challenges faced by deep networks.

Building on the previous analysis, further experiments may involve applying Batch Normalization and Residual Connections separately to VGG08 to understand their individual impact on improving performance in a shallower model that does not face vanishing gradient issues. Subsequently, their combined effect may be examined to determine if integrating these methods leads to better results. The findings can then be compared to the corresponding deep model, VGG38, to assess whether these techniques consistently enhance performance across different architectures, regardless of depth.

Another approach that may improve the model's performance is integrating 1x1 convolutions into downsampling blocks, as suggested by (He et al., 2016b). This adjustment may improve gradient flow and preserve essential information during dimensionality reduction, leading to more robust learning and improved generalization. Another enhancement, based on (He et al., 2016a), is incorporating dynamic learning rates and batch size adjustments to improve training further. Dynamic learning rate stabilizes early training by initially using smaller updates and accelerating convergence. Finally, a batch size of 256 may enhance BN's performance by ensuring stable means and variance estimates, supporting gradient flow.

] .

7. Conclusion

[Question 5 - This study investigates the challenges deeper models such as VGG38 face compared to the shallower VGG08. Deeper networks encounter significant issues such as the vanishing gradient problem and optimization challenges. Batch Normalization (BN) effectively addresses the vanishing gradient problem by improving gradient flow, while Residual Connections (RC) may mitigate optimization difficulties by simplifying learning in deeper layers. Our analysis further highlights that the combination of BN and RC leverages the benefits of both methods, achieving the best performance for VGG38 across all experiments. Additionally, further experiments were proposed to apply BN and RC to the shallower VGG08, where these methods may still enhance training stability and efficiency, even though VGG08 does not face vanishing gradient issues. Techniques such as zero padding and 1x1 convolutional layers were discussed to integrate residual connections into downsampling layers, which may enhance the model's performance. Dynamic learning rate schedules and optimized batch sizes were also explored as methods that could further improve the results.

Based on these findings, a future direction involves integrating dense connectivity as proposed in (Huang et al., 2017). Connecting each layer to all previous layers can complement residual connections by improving feature sharing and gradient propagation. Adding dense connections to the VGG38 architecture may enhance the network's ability to reuse learned features, enabling more efficient learning and potentially achieving better performance by making fuller use of its layers.

]

.

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, 2016a.

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*, 2016b. Specifically refers to Sections 3.2 and 3.3.

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.