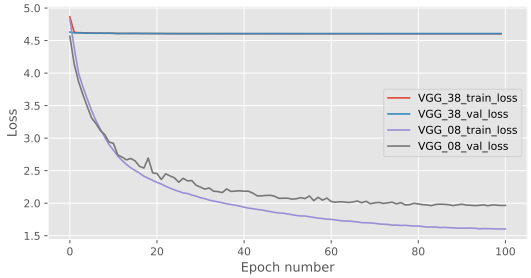
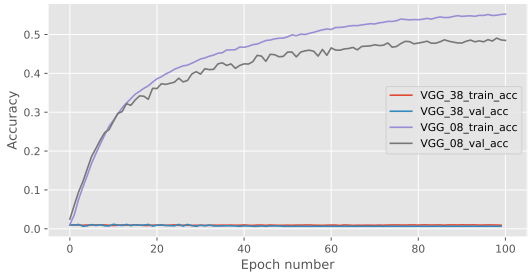
# Combating the Vanishing Gradient Problem in Deep CNNs: A Study of VGG38 with Batch Normalization and Residual Connections


## 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) 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

## 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.

## 2. Identifying training problems of a deep CNN

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

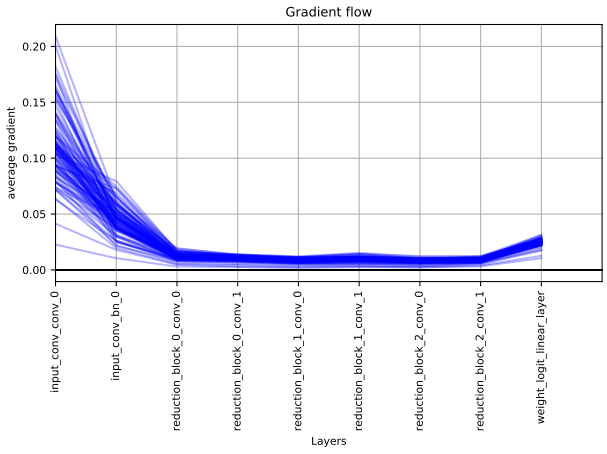



*Figure 2.* Gradient flow on VGG08

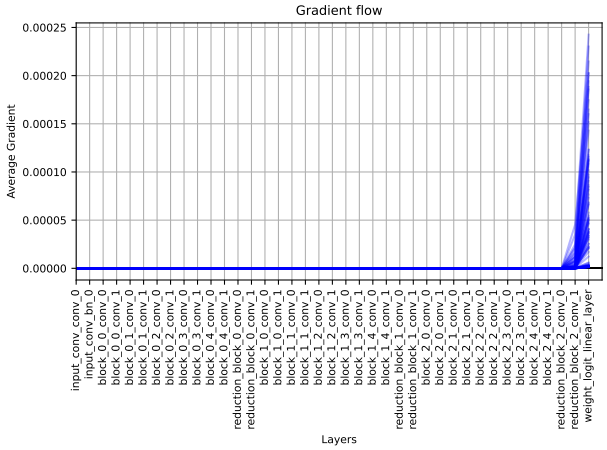



*Figure 3.* Gradient Flow on VGG38

[

]

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. [

The model VGG38 clearly suffers from the Vanishing Gradient problem (VGP). From Figure 1 it can be seen that the VGG38 fails to minimize the loss (both training and validation), since it plateaus at a high value of approximately 4.6 and fails to achieve a higher accuracy, which remains at almost 0 (both training and validation). On the other hand the VGG08 shows more effective learning, with a steady decrease in training and validation loss with an "acceptable" generalization gap and a corresponding increase in accuracy over the epochs.
It can also be seen, by comparing Figures 2 and 3 that the average gradient of the weights of VGG38 is almost 0 in most of the layers, a problem occurring from the first convolution and persists as the gradients propagate through the layers. At the end there seems to be an increase before the fully connected layer but its minimal compared to VGG08, which exhibits a more typical gradient pattern.
The VGP is a significant challenge in gradient-based learning, where small gradients hinder effective weight updates during backpropagation and hamper convergence (He et al., 2016a). This is usually more evident in deeper networks. This prevents the model from learning efficiently, leading to slower (or even stalled) training, reduced accuracy, and poor overall performance (Hochreiter, 1998). To address these consequences, various methods have been developed, such as batch normalization (Ioffe & Szegedy, 2015), ResNet architectures (He et al., 2016a), and novel approaches like dynamically amplifying gradients during backpropagation (Basodi et al., 2020).

] .

## 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.

[

It is commonly believed that increasing network's capacity enables the model to learn more complex patterns in the data. However, this often leads to the issue of overfitting, where the model fits the training data too closely, resulting in very low training errors and poor generalization to new, unseen data (Caruana et al., 2000). On the other hand, (He et al., 2016a) showed that a deeper network with 56 layers had higher training error than a 20-layer network, due to the "degradation problem" where the performance of very deep networks deteriorates. Unlike overfitting, this occurs in models with higher capacity but with higher training errors and no significant generalization gap. Similar findings in (He & Sun, 2015) and (Hashmani et al., 2019) showed stagnant accuracy with increased depth. The issue arises from "optimization issues" in training, including vanishing and exploding gradients which hinder effective weight updates and optimal learning in deeper layers (He et al., 2016a). (Hanin, 2018) linked this performance issue to the lack of architectures specifically designed to mitigate it (such as ResNets, Batch Normalization), improper weight initialization, and the selection of non-linearities (activation functions) that fail to produce numerically stable gradients. ] .

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:

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

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

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 $c, i, j$:

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

where $\gamma \in \mathbb{R}^C$ and $\beta \in \mathbb{R}^C$ are learnable parameters and $\epsilon$ 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

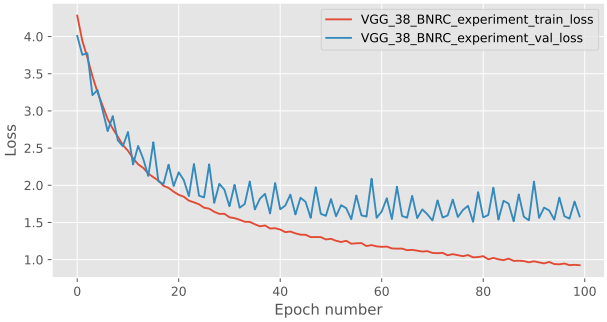



*Figure 4.* Training curves for VGG38 BN+RC (lr =1e-2)

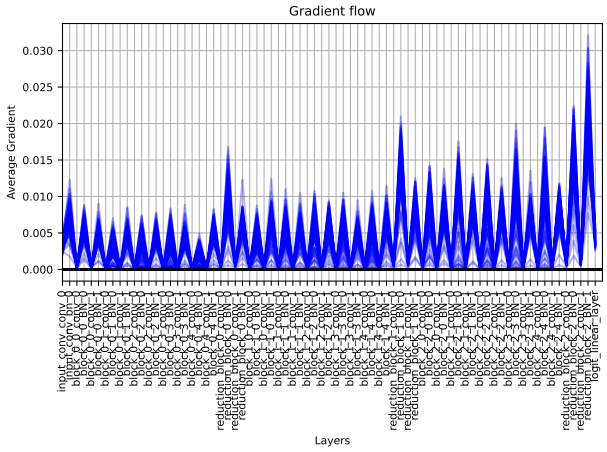



*Figure 5.* Gradient Flow on VGG38 BN+RC (lr =1e-2)

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 $F^{(b)}$ 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

[ ]

[ ]

[

]

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 | 339K | 1.83 | 49.43 | 2.01 | 45.08 |
| 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 | 339K | 0.92 | 71.87 | 1.58 | 59.16 |

*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.

Downsampling introduces challenges because the shortcut in a residual block must align the input and output dimensions; otherwise, the addition (input + output) cannot be performed. The two methods described in (He et al., 2016a), when there is a mismatch in dimensions between the downsampled layer and the input is making the dimensions match either by adding extra padding; or using a 1x1 convolution (with the same stride of downsampling) to match dimensions. Using a 1x1 convolution in the shortcut path offers flexibility and stronger representational abilities, as noted in (He et al., 2016b). In (He et al., 2016b), the authors found that 1×1 convolutional shortcuts are effective in more shallow networks with fewer residual units, enabling input transformation in the shortcut path. However, in very deep networks this did not perform well due to the additional parameters, which increase model complexity and computational cost. These adjustments are applied in the shortcut path (parallel to the main path), and the result is added after the downsampling layer and before the activation function. On the other hand, zero-padding is the simplest and most computationally efficient option but sacrifices some learning capacity, as the zeroes add no trainable parameters or representational power (He et al., 2016a). The padding will be applied in the layer being downsampled to ensure its dimensions match the input dimensions. ].

## 6. Results and Discussion

[

Under the same settings, vanilla VGG08 outperformed VGG38. Despite its higher capacity, VGG38 suffered from VGP (Figure 3), leading to poor learning (0.01 accuracy) and the highest errors, while VGG08 showed better results (with lower training time and parameters). When attempting to combat the VGP for VGG38 with the addition of BN, we can see that the performance improved significantly compared to the vanilla network; for the same learning rate it improves validation loss (2.01 vs. 4.61) and validation accuracy (45.48%). For VGG38 models with BN, higher learning rates improved results, showing BN stabilizes training and enables effective learning with larger weight updates, even in deep networks. When comparing BN and RC we can see for the same configuration that the RC performed better (validation loss 1.84 vs 2.01 and validation accuracy 52.32 vs 45.08). This might be because, BN primarily tackles internal covariate shifts (Ioffe & Szegedy, 2015) and stabilizes the learning by making training more resilient (*indirectly* mitigating the VGP) but the RC *directly* solves the issue by using skip connections that allow gradients to bypass non-linear activations, which can be a cause of VGP in deep networks. A common practice is to combine the BN and RC, which has shown to be beneficial (Thakkar et al., 2018). Indeed, we can see that this network outperformed both the VGG+BN and VGG+RC. Also, a higher learning rate proved beneficial, likely due to the improved gradient flow that allowed the model to handle higher learning rates without overshooting gradients, making it the best-performing model. On the test set this model had an 59.19 test accuracy and 1.57 test loss. Figure 4 shows effective training with decreasing training loss and (fluctuating) declining validation loss. Figure 5, compared to Figure 3, demonstrates that gradients do not vanish, highlighting the architecture's ability to mitigate the VGP.

Further experiments - Observing the improved performance of applying RC compared to BN in this case, and inspired by (He et al., 2016a), where models with depths of 101 and even 152 showed increased performance, we could explore deeper architectures. Since RC effectively addresses challenges in training deeper networks, deeper models are expected to perform at least as well as the current one, if not better. For this a model named VGG70+RC can be run. Also, to better understand and compare the effect of BN vs RC in deeper networks on VGP, we could train a model VGG70+BN and compare it to the aforementioned network, to provide insights into the performance and limitations. Visualizing gradient flow during training in such setups would help further analyze the issues.
Secondly, incorporating RC in the downsampling blocks, using methods like padding or 1x1 convolutions, should further optimize the model's performance and stability since it has shown increased performance in the normal Blocks. These two methods could also be further analyzed to see which one performs better, by contrasting the trade-off between number of parameters (more parameters in 1x1 convolutions) and the learning of the network. For all these models, training could be extended beyond 100 epochs while applying an early stopping criterion (e.g. validation improvement), or even stopped earlier based on Figure 4,

where validation errors fluctuate but plateau (after epoch 40), optimizing both performance and time. Also, increasing the network width by adding more filters per convolutional layer may help address training instability.

] .

## 7. Conclusion

[

Deep Learning faces challenges with deeper networks due to vanishing gradients, which was a major issue in the field. Techniques like BN and RC combat this issue, enabling better training and improved performance in these deeper networks. Research shows convolutional networks can now be deeper, more accurate, and efficient (Huang et al., 2017). The two methods address VGP with different techniques; Batch Normalization (BN) mitigates internal covariate shift by normalizing activations during training while Residual Connections (RC), on the other hand, directly improves the gradient flow by allowing gradients to bypass non-linear layers through skip connections. It was seen that BN stabilized the gradient updates and allowed for a higher learning rate, and also that RC (has a bit more parameters) outperformed BN in this case. Finally, a combination of both these methods combined with a higher learning rate performed the best.
This Future work section would focus on addressing VGP in CNNs, using other well-known methods, that should be used and contrasted with the current networks tested in this paper, based on accuracy/error statistics and by looking at the gradient flow graphs. This could include incorporating methods such as using the "gradient amplification strategy" by (Basodi et al., 2020), which dynamically boosts gradients for specific layers like BN during backpropagation. This could enhance accuracy and support higher learning rates. Future research could also further explore the training time and complexity of the networks. DenseNets (Huang et al., 2017), a network architecture where each layer is connected to all previous layers, ensures efficient gradient flow and feature reuse and therefore requires fewer parameters to alleviate the VGP. Another recommendation is to leverage EfficientNets (Tan & Le, 2019), which optimize depth, width, and resolution scaling using Neural Architecture Search. Incorporating techniques like BN and RC within the EfficientNet framework for a hybrid approach can further enhance gradient flow and stability and overall training robustness.

] .

## References


Basodi, Sunitha, Ji, Chunyan, Zhang, Haiping, and Pan, Yi. Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3): 196–207, 2020.

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.

Caruana, Rich, Lawrence, Steve, and Giles, C. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems*, 13, 2000.

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.

Hanin, Boris. Which neural net architectures give rise to exploding and vanishing gradients? *Advances in neural information processing systems*, 31, 2018.

Hashmani, Manzoor Ahmed, Jameel, Syed Muslim, Alhussain, Hitham, Rehman, Mobashar, and Budiman, Arif. Accuracy performance degradation in image classification models due to concept drift. *International Journal of Advanced Computer Science and Applications*, 10(5), 2019.

He, Kaiming and Sun, Jian. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360, 2015.

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. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645. Springer, 2016b.

Hochreiter, Sepp. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

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.

Tan, Mingxing and Le, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Thakkar, Vignesh, Tewary, Suman, and Chakraborty, Chandan. Batch normalization in convolutional neural networks—a comparative study with cifar-10 data. In *2018 fifth international conference on emerging applications of information technology (EAIT)*, pp. 1–5. IEEE, 2018.