MLP Coursework 2

s2748897

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., 2016b), 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., 2016b). 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., 2016b; 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., 2016b) 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.

2. Identifying training problems of a deep CNN

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:

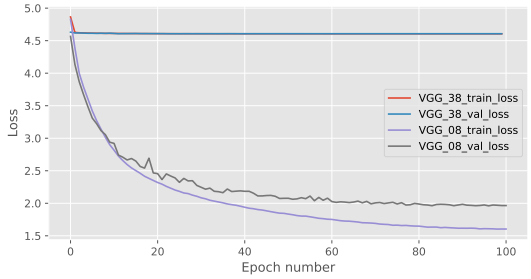
$$\boldsymbol{x}^{(l)} = f^{(l)}(\boldsymbol{x}^{(l-1)}; W^{(l)}) \tag{1}$$

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:

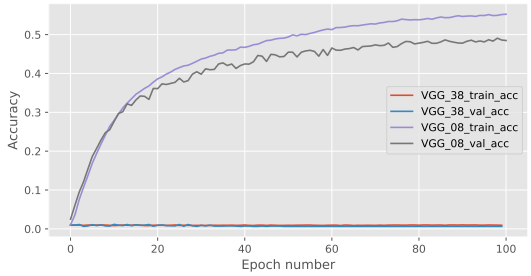
$$\frac{\partial E}{\partial W^{(l)}} = \frac{\partial E}{\partial \boldsymbol{x}^{(L)}} \frac{\partial \boldsymbol{x}^{(L)}}{\partial \boldsymbol{x}^{(L-1)}} \cdots \frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{x}^{(l)}} \frac{\partial \boldsymbol{x}^{(l)}}{\partial W^{(l)}}. \tag{2}$$

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.,



(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

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.

[*Figure 3 inserted*]

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. **[These plots clearly illustrate how the shallower network healthily maintains non-zero gradients during all the epochs, always with reasonably higher values assigned to the first two layers, and consistently lower but significant ones throughout the reduction blocks, slightly increasing back for the final linear layer. The flow is thus relatively stable for the VGG08 model, experiencing no dramatic drop-off or explosion. In contrast, the deeper network demonstrates a severe vanishing gradients issue: in all epochs, the values are uniformly flattened to zero since the early layers, with a light exception of last linear operation's parameters only. In fact, differently from the simpler framework, the VGG38 solution proves to be weak in this setting; its weights are not properly updated across layers, and the learning process becomes inefficient. These observations are also bolstered by figure 1, reporting a satisfactory overall convergence of the VGG08 model to a validation accuracy of 48%, totally outperforming the training-struggling deeper network, which keeps**

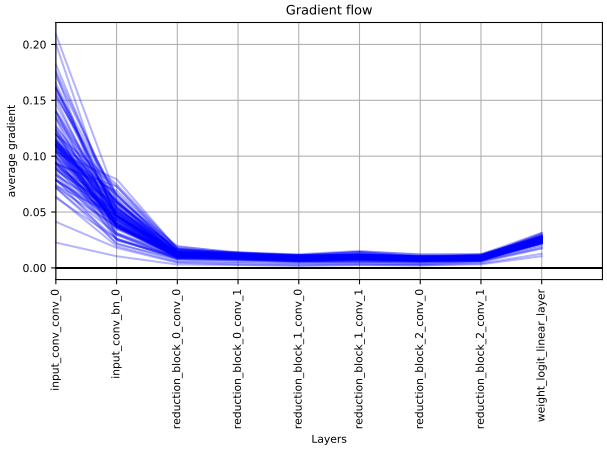


Figure 2. Gradient flow on VGG08

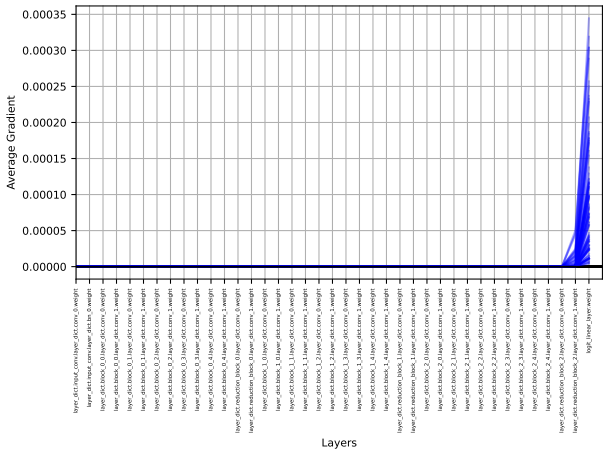


Figure 3. Gradient Flow on VGG38

an unvaried high level of cross entropy after the first weight update, and almost null accuracy values] .

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., 2016b) A well-known way of mitigating the VGP is proposed by He *et al.* in (He et al., 2016b). 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 network capacity – defined as the ability to fit a wide variety of functions – is typically increased for deeper frameworks with respect to shallower ones, due to a higher number of parameters updatable at each epoch. Despite this characteristic can help the model learn more complex data patterns, it also intensifies the risk of overfitting, especially without proper regularization techniques applied. In fact, more powerful networks could tend to capture noise or irrelevant details in not comparably sizeable training data, losing the ability to generalize well to unseen instances. The latter phenomenon would thus imply a high performance on the training set, as opposed by significant flaws with validation and test predictions, which does not fit though the aforementioned scenario; the train accuracy is indeed consistently low for the deeper network, as also in the present study's case. Considering also the equal amount of learning iterations and the previously analysed figure 3, another issue intrinsically related to models' capacity must be impeding training at all: the earlier discussed Vanishing Gradient Problem, of which threat is provenly correlated to both the networks' depth and the activation functions' choice in particular (Ven & Lederer, 2021)] .

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., 2016b; 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 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., 2016b) to tackle the vanishing gradient problem. Introduced by He et. al. (He et al., 2016b), 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., 2016b). 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

[Despite not hereby applied, there are several methods to use residual connections on downsampling layers – usually pooling or convoluting with stride greater than 1 – bypassing the dimensional mismatch between their input and output feature maps. These strategies can be as simple as padding the original feature representation with zeros, allowing again its sum with the downsampling's output one (Han et al., 2016). This slightly modified version of the identity shortcut, which requires no additional parameters, keeps the model simple; however, it might not ensure enough flexibility to capture meaningful information's transformations.

At the cost of computational efficiency, projection shortcuts are better able to learn complex residual mappings, leveraging a 1x1 convolution to project the input feature map into the same dimensions as the downsampled one:

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}, \tag{7}$$

where $F(\mathbf{x}, \{W_i\})$ represents the downsampling transformation, and W_s is the 1x1 convolutional filter projecting x to the appropriate dimensions (He et al., 2016b). This learnable transformation can ensure that the summed feature maps' dimensions are the same without altering the main convolutional operations, introducing new parameters to adapt to different feature representations. As a trade-off, the computational cost is considerably increased, as also the risk of overfitting or hampering information propagation, making the choice between the presented techniques fundamentally related to model's complexity, dataset size, and available hardware evaluations. These common strategies have been extensively discussed also in mixed forms (e.g., adjusting by projection only dimensions' increment, while leaving the remaining shortcuts as identity ones), and tested on CIFAR100 in particular for residual units' validity preservation (He et al., 2016a)] .

6. Results and Discussion

[***Figure 4 inserted*** **]**

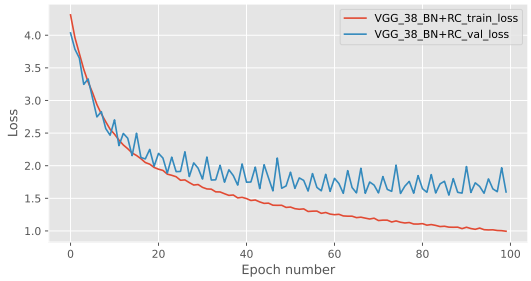
[***Figure 5 inserted*** **]**

[***Table 1 filled*** **]**

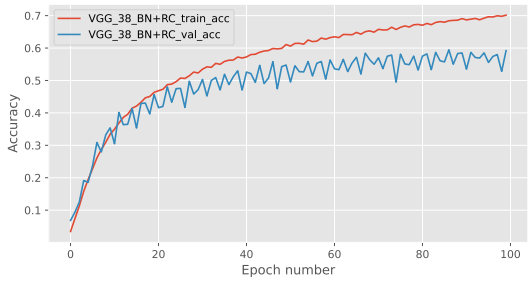
[Table 1 reports the experiment results and demonstrates that, when sustained with Batch Normalization or Residual Connections methods, VGG38 consistently outperforms the shallower network in fitting the training data. In particular, at equal learning rate, Batch Normalization alone allows the deeper model to reach a train accuracy of 1 percentage point superior to VGG08's one, and a validation performance totally

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	1.68	52.67	1.93	46.84
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	1.00	70.15	1.60	59.20

Table 1. Experiment results *(number of model parameters rounded to the thousand, Training and Validation loss and accuracy)* for different combinations of VGG08, VGG38, Batch Normalisation (BN), and Residual Connections (RC) *(LR stands for learning rate)*



(a) Cross entropy error per epoch



(b) Classification accuracy per epoch

Figure 4. Training curves for VGG38 with Batch Normalisation (BN) and Residual Connections (RC) *(learning rate = 1e-2)* in terms of (a) cross-entropy error and (b) classification accuracy

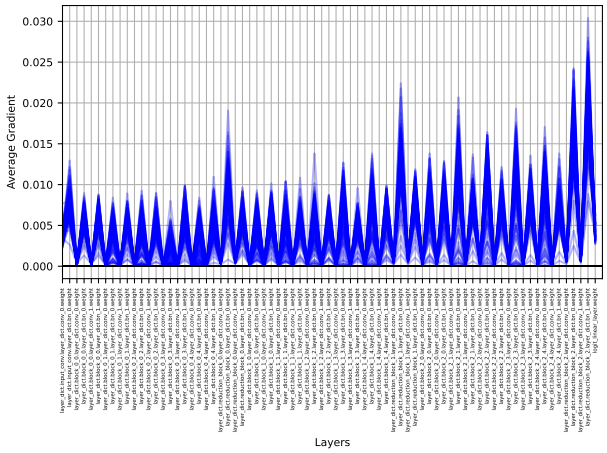


Figure 5. Gradient Flow on VGG38 with Batch Normalisation (BN) and Residual Connections (RC) *(learning rate = 1e-2)*

comparable. However, the use of Residual Connections pushes the network to significantly better learn the training set and get accurate results on 52% of the validation set; the lowest cross entropy errors are though obtained by combining the two strategies, leading to accuracies levels definitely superior with respect to VGG08. While the complex model equipped with BN only reacts negatively to a higher learning rate, blending RC enables VGG38 to well embrace this new parameter, reaching the best results of the series (i.e., the highest train and validation accuracies of 0.70 and 0.59 respectively).

Referring back to the plain deeper network's inconclusive performances, it is immediate to ascertain that the VGP has been hereby solved in each case, as also demonstrated by figure 5. The latter depicts indeed for the best-predicting model an intermittent series of intense peaks of gradients' intensity across almost all the epochs for the Batch Normalization procedures; but most importantly it shows that the convolution layers are not constantly assigned a null correction indication, with the related reduction blocks' average gradient never falling to zero at all. Considering also that the first convolution mostly spans between non-negligible values of 2×10^{-3} and 6×10^{-3}, isolating the core part of the present flow it would be obtained a trend resembling more the shallower network's sound one of figure 2 than the feeble tendency in figure 3. The health of this new framework is also proven by figure 4, presenting the VGG38's training curves as clearly different from the ones in figure 1: the network is now correctly fitting the training set and regularly reaching a good level of accuracy on it, resulting in an overall similar trend of the validation performance – more fluctuating with respect to the original VGG08's one simply because of the deeper model's more complex framework, involving the aforementioned VGP-avoiding techniques too.

To broaden the viewpoint on BN and RC behaviour and virtually further enhance the models' performance, additional experiments might be conducted on the presented networks. Specifically, it might be insightful to apply the incremented learning rate also on VGG38 RC and VGG08 and observe their responses, in order to isolate this augmented parameter's effect on residual con-

nections' contribution and assess whether it could positively influence the shallower network too. The latter result would be unexpected: despite the reduced general propensity of fewer-parameters networks to become unstable at LR's increments, this model's already slightly vacillating training curves (figure 1) suggests that a critical learning-speed level could have been reached yet; it might hence be sensible to apply stronger stabilization methods to VGG08 too before to proceed with higher learning rates. Despite deeper networks are more prone to unhealthy gradient flows and error local-minima's overshooting – due to notable cumulative effects across layers and complex optimization landscapes – VGG38 RC should demonstrate a reduced parameter sensitivity. At LR's increment, a satisfactory convergence is indeed expected for this model and deeper networks leveraging advanced stabilization mechanisms in general, which typically allow larger but thoughtful updates.

In fact, an enhanced performance would also be anticipated when testing even higher learning rates for VGG38 BN+RC; an interesting further experiment would thus consist in tuning the relevant hyperparameter and improve training speed until the convergence stability is verified to default. This could also be applied on BN- and RC-only equipped networks, to empirically distinguish the dependence of these strategies' backing on the chosen LR, but also on disparate new configurations involving the same techniques with different placement along the framework, to fine-tune their application. For instance, Batch Normalization could be tested after activation, which would probably lead to poorer results with the unbounded-output ReLU functions (**Ioffe & Szegedy, 2015**), but might be proper when using sigmoid or tanh instead (**Dubey et al., 2021**), providing a wider understanding of BN's interaction with different architecture types and possibly higher accuracy levels – sacrificing training speed. Different Residual Connections type might also be experimented, leveraging for example the techniques discussed in section **5.1**, and revealing the actual trade-off rate between computational cost and performance improvement by using projection shortcuts instead of zero-padding] .

7. Conclusion

[Through the present experimentation, Batch Normalization and Residual Connections methods have been demonstrated to be pivotal in the training of deeper networks like the analysed VGG38, successfully addressing its Vanishing Gradient Problem. In particular, BN improves the model's stability but shows signs of high sensitivity to the learning rate modification, and using only RC ensures an even higher accuracy in this case; however, combining the two techniques makes the studied architecture reach the best results of the trial set, also at an incremented LR value.

Future work might include more profoundly investigating the effects of these strategies and their synergy at a layer-wise level, adaptively exploring their application to specific sections of the framework, selected as most strongly benefitting from these computationally expensive solutions (e.g., based on a detailed gradient flow analysis, conducted systematically removing BN or RC from different blocks and considering the performance change). Moreover, the interaction of the presented methods with disparate optimizers could be studied, especially testing the less influential ones – Root Mean Square Propagation (RMSProp), or even Stochastic Gradient Descent (SGD) – with a shallower network (e.g., VGG08 itself) on the same dataset, in order to more directly infer the magnitude of the LR choice's influence on such stabilization techniques. Finally, the input information could be varied: leveraging more sophisticated augmentation techniques – like Cutout (**DeVries & Taylor, 2017**), Mixup (**Zhang et al., 2017**), or AutoAugment (**Cubuk et al., 2019**) – more diverse training set could be experimented, ranking possible stratagems to prevent overfitting on limited datasets when models deeper than VGG38 are employed and stabilized. Other datasets at all could instead be useful to evaluate this same network calibre on other tasks than classification (e.g., object detection or segmentation). By delving into the above aspects and building on current findings, more robust and efficient deep learning architectures might be conceivable] .

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.

Cubuk, Ekin Dogus, Zoph, Barret, Mané, Dandelion, Vasudevan, Vijay, and Le, Quoc V. Autoaugment: Learning augmentation strategies from data. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 113–123, 2019. URL https://api.semanticscholar.org/CorpusID:196208260.

DeVries, Terrance and Taylor, Graham. Improved regularization of convolutional neural networks with cutout. 08 2017. doi: 10.48550/arXiv.1708.04552.

Dubey, Shiv Ram, Singh, Satish Kumar, and Chaudhuri, Bidyut Baran. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2021. URL https://api.semanticscholar.org/CorpusID:250089226.

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.

Han, Dongyoon, Kim, Jiwhan, and Kim, Junmo. Deep pyramidal residual networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6307–6315, 2016. URL https://api.semanticscholar.org/CorpusID:5398883.

He, Kaiming, Zhang, X., Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016a. URL https://api.semanticscholar.org/CorpusID:6447277.

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, 2016b.

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.

Ven, Leni and Lederer, Johannes. Regularization and reparameterization avoid vanishing gradients in sigmoid-type networks. *ArXiv*, abs/2106.02260, 2021. URL https://api.semanticscholar.org/CorpusID:235352991.

Zhang, Hongyi, Cissé, Moustapha, Dauphin, Yann, and Lopez-Paz, David. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412, 2017. URL https://api.semanticscholar.org/CorpusID:3162051.