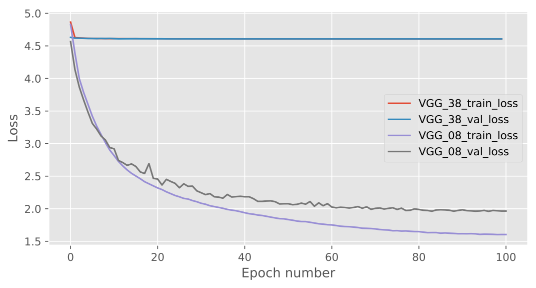
MLP Coursework 2

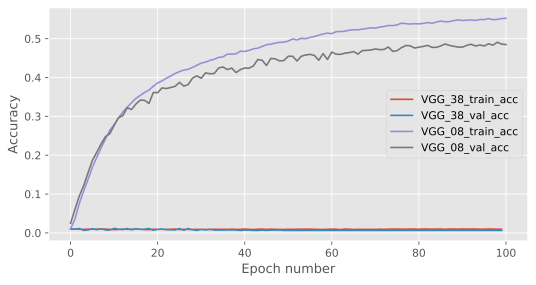
Kyriakos Kyriakou s2281922

Abstract

Deep neural networks have become the state-of-the-art in many standard computer vision problems thanks to their powerful representations and availability of large labeled datasets. While very deep networks allow for learning more levels of abstractions in their layers from the data, training these models successfully is a challenging task due to problematic gradient flow through the layers, known as vanishing/exploding gradient problem. In this report, we first analyze this problem in VGG models with 8 and 38 hidden layers on the CIFAR100 image dataset, by monitoring the gradient flow during training. We explore known solutions to this problem including batch normalization or residual connections, and explain their theory and implementation details. Our experiments show that batch normalization and residual connections effectively address the aforementioned problem and hence enable a deeper model to outperform shallower ones in the same experimental setup.



(a) Loss per epoch



(b) Accuracy per epoch

Figure 1. Training curves for VGG08 and VGG38

1. Introduction

Despite the remarkable progress of deep neural networks in image classification problems (Simonyan & Zisserman, 2014; He et al., 2016), training very deep networks is a challenging procedure. One of the major problems is the Vanishing Gradient Problem (VGP), a phenomenon where the gradients of the error function with respect to network weights shrink to zero, as they backpropagate to earlier layers, hence preventing effective weight updates. This phenomenon is prevalent and has been extensively studied in various deep neural networks including feedforward networks (Glorot & Bengio, 2010), RNNs (Bengio et al., 1993), and CNNs (He et al., 2016). Multiple solutions have been proposed to mitigate this problem by using weight initialization strategies (Glorot & Bengio, 2010), activation functions (Glorot & Bengio, 2010), input normalization (Bishop et al., 1995), batch normalization (Ioffe & Szegedy, 2015), and shortcut connections (He et al., 2016; Huang et al., 2017).

This report focuses on diagnosing the VGP occurring in the VGG38 model and addressing it by implementing two standard solutions. In particular, we first study a "broken" network in terms of its gradient flow, norm of gradients with respect to its weights for each layer and contrast it to ones in the healthy VGG08 to pinpoint the problem. Next, we review two standard solutions for this problem, batch normalization (BN) (Ioffe & Szegedy, 2015) and residual connections (RC) (He et al., 2016) in detail and discuss how they can address the gradient problem. We first incorporate batch normalization (denoted as VGG38+BN), residual connections (denoted as VGG38+RC), and their combination (denoted as VGG38+BN+RC) to the given VGG38 architecture. We train the resulting three configurations, and VGG08 and VGG38 models on CIFAR-100 dataset and present the results. The results show that though separate use of BN and RC does mitigate the vanishing/exploding gradient problem, therefore enabling effective training of the VGG38 model, the best results are obtained by combining both BN and RC.

2. Identifying training problems of a deep CNN

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 previ-

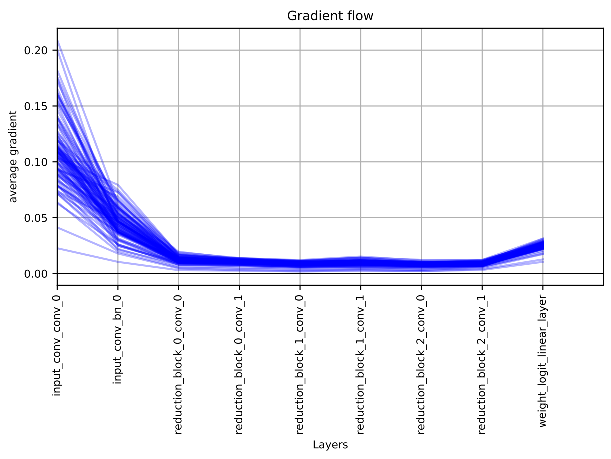


Figure 2. Gradient flow on VGG08

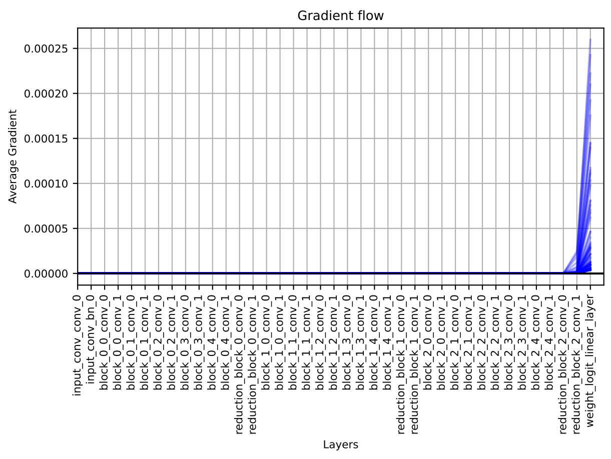


Figure 3. Gradient Flow on VGG38

ous 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:

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

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. Figure 1 (training curves) can be used to identify the Vanishing Gradient Problem. The problem can be identified on both loss and accuracy graphs in the model VGG38. The model cannot improve its accuracy and reduce the loss through the training epochs and that is shown with a flat line. Figures 2 and 3 with the gradient flows, can be also used to identify the issue. The VGG08 model gradients are shown to be healthy as we go backward with every layer. This is because the derivatives are not only getting vanishingly smaller, but also increase as well during back-propagation. On the other hand, the gradients of the unhealthy VGG38 model, continue to get smaller and smaller as we go backwards. Eventually the gradients are vanished. Consequently, the model training takes longer, or even fails to train and learn new patterns as the weights are not updated properly.

3. Background Literature

In this section we will highlight some of the most influential papers that have been central to overcoming the VGP in deep CNNs.

Batch Normalization (Ioffe & Szegedy, 2015) BN seeks to solve the problem of internal covariate shift (ICS), when distribution of each layer's inputs changes during training, as the parameters of the previous layers change. The authors argue that without batch normalization, the distribution of each layer's inputs can vary significantly due to the stochastic nature of randomly sampling mini-batches from your training set. Layers in the network hence must continuously adapt to these high variance distributions which hinders the rate of convergence gradient-based optimizers. This optimization problem is exacerbated further with network depth due to the updating of parameters at layer l being dependent on the previous $l-1$ layers.

It is hence beneficial to embed the normalization of training data into the network architecture after work from LeCun *et al.* showed that training converges faster with this addition (LeCun et al., 2012). Through standardizing the inputs to each layer, we take a step towards achieving the fixed distributions of inputs that remove the ill effects of ICS. Ioffe and Szegedy demonstrate the effectiveness of their technique through training an ensemble of BN networks which achieve an accuracy on the ImageNet classification task exceeding that of humans in 14 times fewer training steps than the state-of-the-art of the time. It should be noted, however, that the exact reason for BN's effectiveness is still not completely understood and it is an open research question (Santurkar et al., 2018).

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

Deeper neural networks that have bigger capacity tend to lead to the oveffiitting problem more as the networks tend to memorize specific patterns only. Figure 1 from (He et al., 2016) depicts that the degradation of the bigger network does not come from overfitting. The 56-layer, which has bigger capacity, has constantly higher training error throughout training than the 20-layer network with smaller capacity. In the case the model overfits, the 56-layer network would have showed overfitting throughout the testing with the testing error getting worse after iterations (entering overfitting region). At the same time, the training error of the deeper model would have kept getting smaller, something that is not happening on Figure 1 as the shallow 20-layer network has smaller training error. The degradation problem that has been exposed by the authors on Figure 1 of (He et al., 2016) is clearly not coming from overfitting, but instead, due to the Vanishing/Exploding Gradients Problem.

Residual networks, colloquially known as ResNets, aim to alleviate VGP through the incorporation of skip connections that bypass the linear transformations into the network architecture. The authors argue that this new mapping is significantly easier to optimize since if an identity mapping were optimal, the network could comfortably learn to push the residual to zero rather than attempting to fit an identity mapping via a stack of nonlinear layers. They bolster their argument by successfully training ResNets with depths exceeding 1000 layers on the CIFAR10 dataset. Prior to their work, training even a 100-layer was accepted as a great challenge within the deep learning community. The addition of skip connections solves the VGP through enabling information to flow more freely throughout the network architecture without the addition of neither extra parameters, nor computational complexity.

4. Solution overview

4.1. Batch normalization

Batch Normalization is an effective way to address the vanishing gradient problem. The authors in (Ioffe & Szegedy, 2015) introduce Batch Normalization Transform which is used to deal with their problem of Internal Covariate Shift. The same algorithm proposed by Loffe and Szegedy is used for the VGP. The BN algorithm take a mini-batch as input and two parameters to be learned (β and γ). The output of the algorithm is a normalized batch. The Batch Normalization formula defined in (Ioffe & Szegedy, 2015) is:

$$BN_{\gamma,\beta} : x_{1...m} \rightarrow y_{1...m} \tag{3}$$

The normalization consists of four steps during training time:

1. Calculating the mini-batch mean: $\mu_{\mathcal{B}}$
2. Calculating the mini-batch variance: $\sigma^2_{\mathcal{B}}$
3. Normalize the mini-batch: $\hat{xi} = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$
4. Lastly, scale and shift with the learning parameters β and γ: $y_i \leftarrow \gamma \hat{x}_i + \beta$

During testing time the normalization of activations is not desirable as we would like the output to be established only on the input. Therefore only the population is used for testing instead of the mini-batches. Step **3** is modified to: $\hat{x} = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$.
BN is an effective way to resolve the VGP because by normalizing the activations in the deep network, not big changes of the parameters are allowed to be made while going through the layers (Ioffe & Szegedy, 2015). Consequently, BN will keep the layer parameters steady which will not allow the gradients to vanish or explode.

4.2. Residual connections

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

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

$$\frac{\partial \boldsymbol{x}_{(b)}}{\partial \boldsymbol{x}^{(b-1)}} = \mathbb{1} + \frac{\partial F(\boldsymbol{x}^{(b-1)})}{\partial \boldsymbol{x}^{(b-1)}} \tag{4}$$

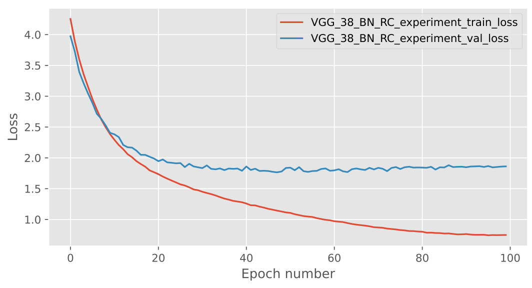
where $\boldsymbol{x}^{(b-1)} \in \mathbb{R}^{H \times W \times C}$ and $\mathbb{1}$ is a $\mathbb{R}^{H \times W \times C}$-dimensional tensor with entries 1. Importantly, $\mathbb{1}$ prevents the zero gradient flow.

5. Experiment Setup

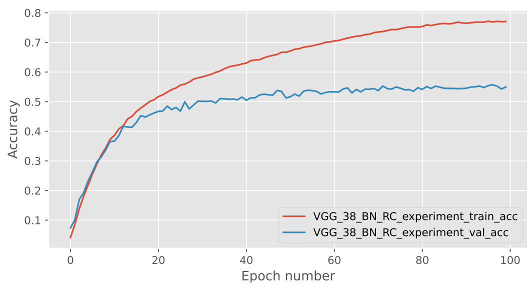
We conduct our experiment on the CIFAR-100 dataset (Krizhevsky et al., 2009), which consists of 60,000 32x32 colour images from 100 different classes. The number of

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	1.55	55.92	2.14	43.52
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	0.75	77.08	1.86	54.96

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.



(a) Loss per epoch



(b) Accuracy per epoch

Figure 4. Training curves for best model - VGG38 with Batch Normalization and Residual Connections (learning rate of 1e-2)

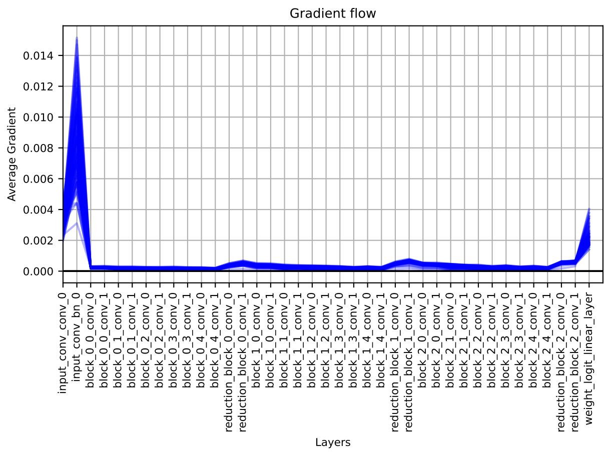


Figure 5. Gradient Flow on best model - VGG38 with Batch Normalization and Residual Connections (learning rate of 1e-2)

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 Figure 4.

When used, BN is applied after each convolutional layer, before the Leaky ReLU non-linearity. Similarly, the skip connections are applied from before the convolution layer to before the final activation function of the block as per Figure 2 of (He et al., 2016). Note that adding residual connections between the feature maps before and after downsampling requires special treatment, as there is a dimension mismatch between them. Therefore in the coursework, we do not use residual connections in the down-sampling blocks. However, please note that batch normalization should still be implemented for these blocks.

5.1. Residual Connections to Downsampling Layers

Applying residual connections to the downsampling layers was not incorporated because a transformation is need to the input feature to map the new downsampled dimension. Our implementation is using an element-wise addition of the mapping and its input feature map that do have the same dimensions (identity mapping). A replace of the input feature map is needed to transform everything to the same dimensions. Adding a linear transformation to the input feature map to match the dimension can solve this problem (He et al., 2016). Therefore, the change needed in our RC implementation to apply RC in downsampling layers (projection mapping) would be $\boldsymbol{x}^{(b)} = W\boldsymbol{x}^{(b-1)} + F(\boldsymbol{x}^{(b-1)})$ where $\boldsymbol{W}$ is the transformation matrix. Two ways to incorporate these two residual connections applications is, firstly, to use the projection shortcuts when dimensionality reduction blocks are used (downsampling layers) and use the identity projections (without linear transformation) in the other layers. A second way to incorporate the changes is to apply to all shortcuts only projection mappings. These two incorporations are been tested on the (He et al., 2016). Their results show that applying only projection mappings, the model achieves a slightly better generalization than applying with the first way. Applying projection shortcuts only, even though that the model generalizes better (learns more parameters), it's increasing the model size by a lot and it's not memory and time efficient. On the other hand incorporating the first way (identity mapping + projection mapping), it might get slightly worse accuracy, but the model is much more efficient in terms of model size and memory complexity. Therefore, this incorporation is an overall better approach implementing RC with downsampling layers (He et al., 2016).

6. Results and Discussion

In the experiments, a shallow model (VGG08) was used along with other deeper models (VGG38) with many variations that include adding BN, RC, both, and changing the learning rate. Based on the experiments carried out, the best performing model is the VGG38 with batch normalization and residual connections and with learning rate of 1e-2, achieving a validation accuracy of 54.96%. Even though the model achieved the best generalization is worth noting that it was on the edge of overfitting as the cap of training accuracy and validation accuracy is big with 22.12% (Figure 4 – a)). Furthermore, in the Figure 4 – b), a gap in the loss can be also seen where at the epoch 80, the loss starts to increase a slightly as well indicating overfit. Adding batch normalization to the deep models showed that it increases the learning parameters of the model (from 336K to 339K), thus these models obtained a higher training accuracy compared to the shallow one. That's not the case for the generalization accuracy though as the shallow model outperformed the deeper with BN. Increasing the learning rate for that model (from 1e-3 to 1e-2) showed an increase on the validation accuracy, but still, wasn't enough to perform better than the VGG08. Another model tested was a VGG38 model with the residual connections technique applied. The results showed that this model outperformed the shallowed model getting better generalization scores. Also, the model with RC even though it obtained less learning parameters than the model with BN, did train and generalize better too. Finally two more experiment were conducted, applying BN + RC together with two different learning rates (1e-3 and 1e-2). These experiments showed that the models were performing better with bigger learning rate and also a combination of BN + RC in the deeper models not only tackle the VGP, but help improve the generalization of the models as well.

Further future experiments could be run in order to get better insights on the behaviour of BN and RC, and see how we could improve the performance of the already trained model. A good experiment would be to run the best performing model (VGG38 with BN+RC), but include RC to the downsampling layers as well, and see what impact has that on the model. This can be also used to understand more how residual connections work the best in the model (identity/projection mapping). Another good experiment to see how BN behaves is to use the shallow VGG08 model and apply BN on it. That could give a good indication if BN can also help really with the generalization of the model and not only with the VGP. Finally, we could try increasing the learning rate of our best performing model to see if an even bigger learning rate can increase the performance of the model. These are good experiments that could be used to draw more detailed conclusions.

7. Conclusion

While the vanishing gradients problem is known to run in very deep networks, in this study we analyzed it in the VGG models with 8 and 38-hidden layers on the CIFAR-100 image dataset. We examined previous literature knowledge on known solutions for this problem, including batch normalization and residual connections, and, in our experimental study we investigated the effect of these two techniques on different networks. The findings of this study showed that both batch normalization and residual connections addressed effectively the problem and hence enabled deeper models to perform better than more shallowed ones. Furthermore, we concluded that the best performing model in our experiments was a model that combined both batch normalization and residual connections techniques. Finally, we discussed future incremental work with possible new combinations to improve model performance and get more insights about the behaviour of BN and RC. There remains theoretical questions to be considered, such whether the overfitting observed in the best performing model can be tackled with the introduction of different regularization methods such as dropout and weight decay.

References

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

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

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

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

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

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

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

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

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

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

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

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