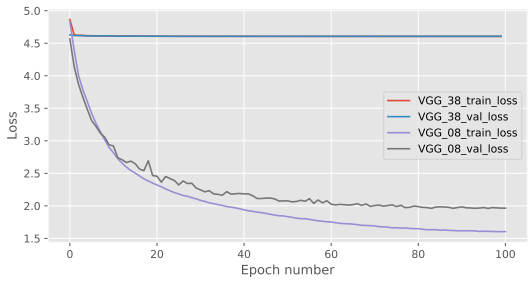
MLP Coursework 2

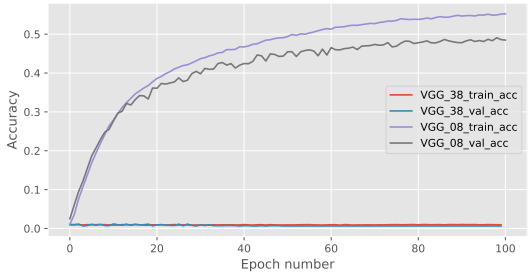
s1802092

Abstract

Deep neural networks have become the state-of-the-art in many standard computer vision problems thanks to more powerful neural networks and large labeled datasets. While very deep networks allow for better deciphering of the complex patterns in the data, training these models successfully is a challenging task due to problematic gradient flow through the layers, known as vanishing/exploding gradient problem (VGP and EGP respectively). 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 VGP, a phenomenon where gradients from the loss function shrink to zero as they backpropagate to earlier layers, hence preventing the network from updating its weights effectively. This phenomenon is prevalent and has been extensively studied in various deep network including feed-forward 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 occurred in the VGG38 model and addressing it by implementing two standard solutions. In particular, we first study the "broken" network in terms of its gradient flow, norm of gradients with respect to model 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 tackle the vanishing/exploding gradient problem, therefore enabling the training of the VGG38 model, the best results are obtained by combining both BN and RC.

2. Identifying training problems of a deep CNN

[Question Figure 3 - Replace this image with a figure depicting the average gradient across layers, for the VGG38 model.]

Concretely, training deep neural 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 x^0 to the network and

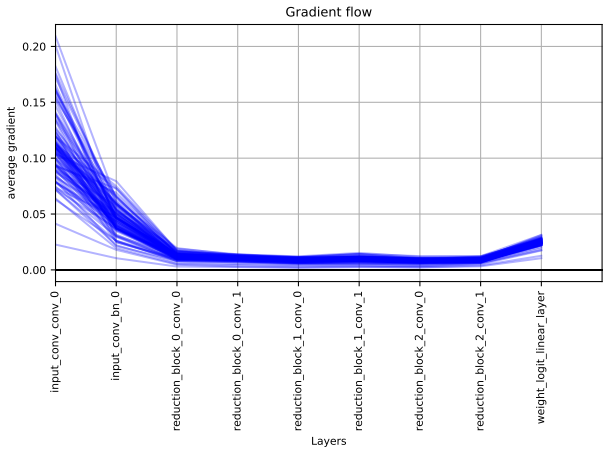


Figure 2. Gradient flow on VGG08

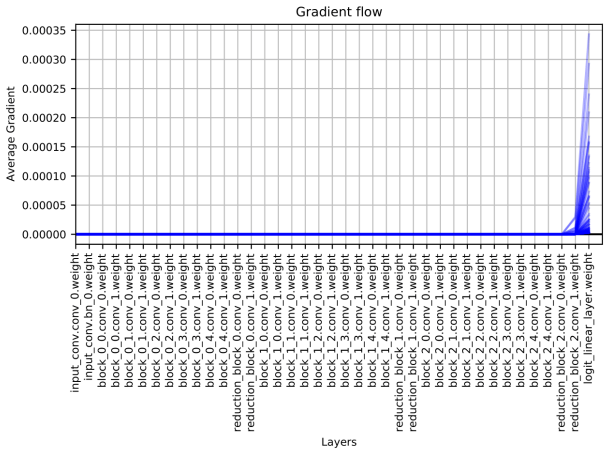


Figure 3. Gradient Flow on VGG38

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:

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

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:

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

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. **[Looking at the training curves for VGG08 and VGG38 in Figure 1, we observe that both the loss and the accuracy corresponding to VGG38 remains constant (fail to converge to a desriable value) beyond the initial epoch when evaluated on both the training and validation sets whereas the loss and accuracy for VGG08 decreases and increases respectively with epoch number. This means that learning does not occur or is at an extremely slow rate for the VGG38 model, and We can further identify the cause by comparing the gradient flows for both models. For the VGG08 model, it is clear from Figure 2 that the mean absolute gradient values for he weights do not attain 0 as we backpropagate from the last layer to the first layer (read from right to left of the graph). Recall that the weights are updated using $w^{'} = w - \eta * gradient$ where η is the learning rate. Since gradients are never zero for each layer, we successfully update the weights during training time to minimise the loss. In contrast to this, Figure 3 depicts the gradient flow for the VGG38 model whereby the mean absolute gradient values are 0 across all layers except for the final two layers. After backpropagating for two layers, the gradients for preceding layers 'vanish' perhaps because of extremely small partial gradient terms which are used for computing them in the chain rule as shown in equation (2). Therefore, the weights in these initial layers remain constant (not updated) throughout training which in turn results in no change in the loss function value. We can conclude that the model VGG38 suffers from the VGP as described previously]** .

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 con-

tinuously 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) One interpretation of how the VGP arises is that stacking non-linear layers between the input and output of networks makes the connection between these variables increasingly complex. This results in the gradients becoming increasingly scrambled as they are propagated back through the network and the desired mapping between input and output being lost. He *et al.* observed this on a deep 56-layer neural network counter-intuitively achieving a higher training error than a shallower 20- layer network despite higher theoretical power. Residual networks, colloquially known as ResNets, aim to alleviate this 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 Normalisation (BN) is able to alleviate the VGP. By normalising (standardising), shifting and scaling the inputs x_i to each layer or activation, we ensure that $|\ x_i\ |$ no longer react strongly to small changes in parameters and is within an appropriate range such that activation functions like logistic sigmoid or tanh do not saturate, preventing extremely small (vanishing) gradients which leads to slow convergence. The steps for implementing BN can be summarised using the equations below:

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

$$\begin{aligned} z_i &= \gamma_i \hat{u}_i + \beta_i \\ &= batchNorm(u_i) \end{aligned} \qquad (4)$$

where u_i is a hidden unit activation $u_i = \mathbf{w}^T x_i$, and ϵ is a small float added to variance to avoid dividing by zero.Based on the equations, we first standardise each u_i using its sample mean and variance of each computed across a minibatch of size M:

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

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

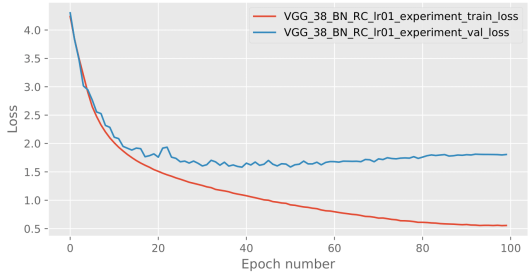
We then apply the $bathNorm()$ function, essentially shifting and scaling the standardised hidden units $\hat{u}_i$ with parameters γ_i and β_i as show in equation (4), which can be learned during training. This is to ensure that the stadardised activations have new means and standard deviations to prevent them from having values that are too small to 'activate' the activation functions, thus maintaining the expressive power of the network. The steps just described are applicable for implementing BN during training time. During test time, the same can be said except for the sample mean and variance used since we only have a single sample but not minibatches. In this case, the mean and variance for each layer are obtained from the mean and variance over the full training data.

] .

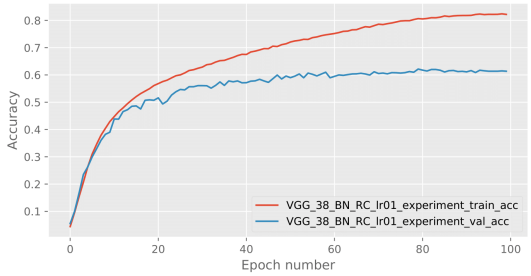
4.2. Residual connections

[Question 3 - Describe Residual Connections (RC) by using equations. Consider both training and test time and explain how RC address the vanishing gradient problem. In addition to BN, Residual Connections (RC) which are part of the ResNet architecture are also able to tackle the VGP. RC builds on the idea of skip (shortcut) connections whereby the output of one layer is fed as the input to the other layers deeper in network instead of just the next layer. This provides an alternative path or a 'highway' for gradients to flow by skipping some layers in between ensuring that they do not vanish to 0 during backpropagation to initial layers. To implement RC, we simply reformulate the stacked layers (block) as learning residual functions $F(\mathbf{x}, \{W_i\})$ instead of approximating the original underlying mapping $H(x)$ like how traditional networks do. Letting $\mathbf{x}$ be the input to the stacked layers, we have that:

$$F(\mathbf{x}, \{W_i\}) = H(\mathbf{x}) - \mathbf{x} \qquad (7)$$



(a) Loss per epoch



(b) Accuracy per epoch

Figure 4. Training curves for VGG38 BN+RC with learning rate 1e-2

We rearrange to obtain:

$$H(\mathbf{x}) = F(\mathbf{x}, \{W_i\}) + \mathbf{x} \qquad (8)$$

We observe in the above equations that we have an identity mapping coming from x, therefore there are no extra parameters to be learned, and the layers are essentially learning the residual $F(\mathbf{x}, \{W_i\})$ which has the added advantage of being easier to optimise compared to the original mapping $H(\mathbf{x})$. Note that the equations above do not account for when $F(\mathbf{x}, \{W_i\})$ and the input vector x have different dimensions. In this case, we can perform a linear projection on x , i.e. W_s to match the dimensions:

$$H(\mathbf{x}) = F(\mathbf{x}, \{W_i\}) + W_s\mathbf{x} \qquad (9)$$

However, in our subsequent experiments involving RC, we only applied RC to blocks without dimensionality reduction, meaning that the inputs to a block x will have the same dimension as $F(\mathbf{x}, \{W_i\})$] .

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 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

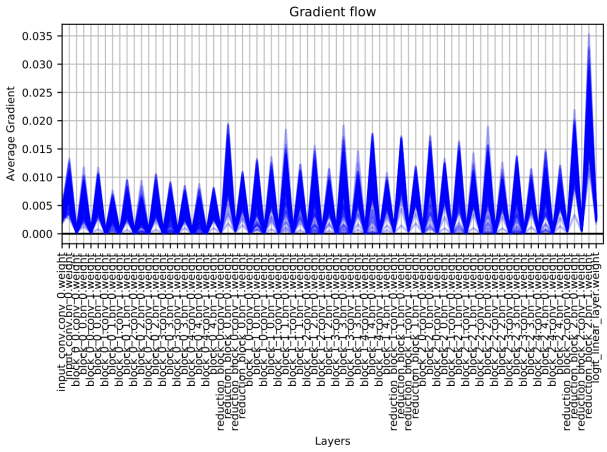


Figure 5. Gradient Flow on VGG38 BN+RC with learning rate 1e-2

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 1.

When used, BN is applied after each convolutional layer, before the Leaky ReLU non-linearity. Similarly, the skip connections are applied from before the convolution layer to before the final activation function of the block as per Figure 2 of (He et al., 2016)

6. Results and Discussion

[Based on Table 1, we observe that the performance of the shallower network VGG08 is significantly better than its deeper counterpart VGG38 which appears to have no predictive power at all given its 0.01% accuracy. This has been previously discussed as being caused by the VGP that deeper networks tend to suffer from. Therefore, subsequent experiments that incorporated BN and RC are conducted to improve on the performance of VGG38.

We first look at the results for when we applied BN with the initial default learning rate of 1e-3, (VGG38 BN 1e-3). As expected, the model's performance has improved significantly, even attaining comparable results to the VGG08 model with its higher training accuracy and slightly lower validation accuracy (only by 0.12%). This implies that we have addressed the issue of vanish-

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	1.57	55.46	1.98	46.72
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	0.56	82.19	1.81	61.36

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.

ing gradient using BN. According to (**Ioffe & Szegedy, 2015**), BN allows for higher learning rate to be used since it reduces the dependence of gradients on the scale of the parameters. Therefore, we further experimented by applying BN with an initial learning rate of 1e-2. However, it seems that there is no improvement in terms of the validation accuracy - both VGG38 BN using the different initial learning rates have validation accuracy of 46.72%, whereas the training accuracy when using 1e-3 is even better (55.46%) than that when using 1e-2 (52.28%). However, this may be due to the fact that we are using a cosine annealing learning rate schedule but not a constant learning rate. Therefore, in order to observe the true effects of learning rate, perhaps experiments applying BN with different constant lerning rates should be conducted.

Next, we look at the results for when RC is being applied with the initial default learning rate of 1e-3, (VGG38 RC 1e-3). It appears that the performance in terms of the trainiing and validation loss and accuracy are all better than when we apply BN. In fact, its validation accuracy is 5.60% higher than that of the model using BN. This may indicate that RC is better at tackling the VGP in deep networks. However, we observe that the generalization gap (Val loss – Train loss) for VGG38 BN 1e-3 (0.41) is lower than that of VGG38 RC 1e-3 (0.51). This may be because of the added advantage of how BN can act as a regulariser, thus reducing the need for other regularization techniques (**Ioffe & Szegedy, 2015**). Again, to truly observe this fact, further experiments could have been conducted, e.g. experiments comparing models using only BN with models using both BN and other regularization techniques like dropout or weight penalties. As mentioned previously, RC is currently being applied in blocks without dimensionality reduction which maintains the same dimensions across the inputs and outputs and thus relies only on the identity mapping of x (refer to equation (8)). Since it has been shown by (**He et al., 2016**) that projection shortcuts (refer to equation (9)) lead to slightly better performance since extra parameters are introduced, we can therefore consider including RC in dimensionality reduction blocks as an attempt to improve performance.

Finally, we experimented with models combining both BN and RC and observe that they outperformed all the previous models which used BN and RC independently. In particular, the model using the higher initial learning rate of 1e-2 (VGG38 BN+RC 1e-2) gave the best overall performance, whereby its training curves and gradient flow plots are shown in Figures 4 and 5. Comparing its gradient flow plot to the one obtained for VGG38 in Figure 2, we can clearly see that the average gradients are non zero for VGG38 BN+RC 1e-2 as they backpropagate, indicating that the VGP has been well addressed. Unlike when we only applied BN, changing the initial learning rate has a large impact on the results when combining BN and RC, and the resulting generalisation gap for VGG38 BN+RC 1e-2 is relatively large (1.25) compared to the other models. This could imply that although this model is best-performing model, it could be overfitting to the training set. Therefore, there is still potential for it to achieve better generalisaion and thus better results and this could be further investigated by using other regularisation techniques.

When comparing between VGG38 BN and VGG38 BN+RC, we observe there is significant improvement in performance in the latter model. Based on the experimental setup in (**He et al., 2016**), it is argued that such an improvement is not attributed to solving the VGP since it has already been alleviated by BN. Instead, by adding RC to VGG38 BN, we may have tackled another issue prevalent in deep networks known as the degradation problem, which occurs when increasing the depth of a network leads to a decrease in performance on both test and training data. This could explain why we have improved performance when combining BN and RC.

With VGG38 BN+RC 1e-2 being the best performing model, we further evaluated its performance on the test set and obtained a loss of approximately 1.74 and an accuracy of 60.69%, which is not too different from the results on the validation set] .

7. Conclusion

[When training deep convolutional neural networks, issues like the vansihing gradient problem (VGP) tend to arise, resulting in an inability to learn the weights effectively. We therefore proposed methods like Batch

Normalisation (BN) and Residual Connections (RC) as solutions to the VGP. Through experiments comparing VGG38 models that incorporated these methods against the original VGG38 baseline, we showed that BN and RC are indeed successful in tackling the VGP. In fact, the best performance is obtained when combining both methods which result in a test accuracy of 60.69%.

However, we do believe that this can be further improved by exploring other methods, apart from increasing the network depth, fine tuning the learning rate and trying different regularization techniques. For instance, by using Dense Convolution Network (DenseNet) which is another form of skip connection, as proposed by (Huang et al., 2017). In DenseNet each layer receives weights from all of its preceding layers by concatenating the preceding layer's output feature maps with the next layer. As a result, shorter paths are created leading to maximise information flow as features extracted early layers are directly used by deeper layers, which would be helpful for addressing both the VGP and degradation problem. Furthermore, there are less parameters and we can train these networks with higher computational and memory efficiency compared to our current methods] .

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.