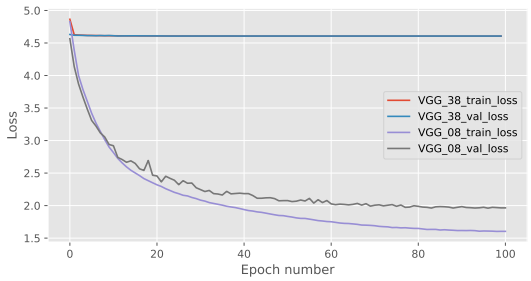
MLP Coursework 2

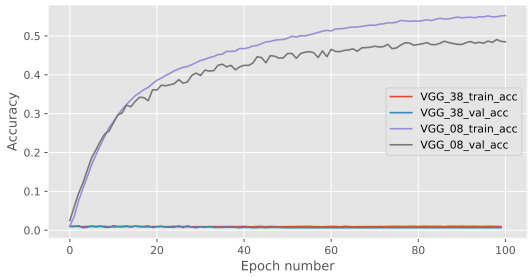
s1720422

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

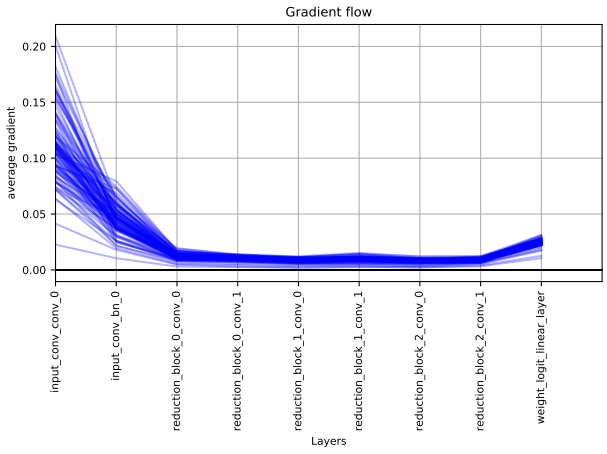


Figure 2. Gradient flow on VGG08

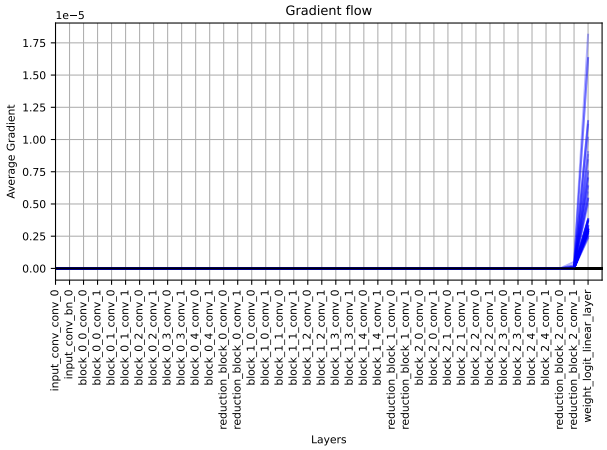


Figure 3. Gradient Flow on VGG38

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:

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

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. **[It can be observed from figure 1 that the VGG08 model depicts the expected training and test accuracy/error of a healthy CNN. The figure shows that over 100 epochs the training/test accuracy increases at a sginificant rate to a final value of 55.2% and 48.5% respectively. Furthermmore, over the period of 100 epochs we can observe from the error curves that the genrealization gap for the model is relatively small and the error curves consistently decrease, thus overfitting is unlikely to be occuring. In contrast, over 100 epochs it can be observsed that the VGG38 model perfromed poorly. The training/test accuracies virtually remained at approximately 0 thorughout the training of the model. Correspondingly, the trainig/test error for the VGG38 model reamined at approximately 4.6 to 4.8 thorugout the 100 epochs and failed to decrease unlike the 8 layer model.**

Figures 2 and 3 highlight that the disparity in performance between both models is due to the Vanishing Gradient Problem. The gradient flow plot produced in figure 2 portrays the expected range of backpropogated gradients across 100 epochs for a healthy CNN. This is due to the variation in gradient over every epoch still remaining sginificant within the first few layers as observed by the average gradeint in early layers fluctuating approximately between 0.025 and 0.20. Converssely, figure 3 shows that there is barely any gradient variation in the majority of layers of the VGG38 model with the gradient approximately being near 0 for the majority of the time. The plot demonstrates the vanishing gradient problem caused by backpropgated gradieents converging towards 0. It is a result of gradients for the leaky ReLu activation function being less than 1 which is constantly being multiplied thorughout the depth of the network, hence causing computed gradeints to converge towards 0. The consequencee of having near 0 gradeints in most layers is a model which predicts target outputs very pporly due to not being able to learn features within the inputs very well. The gradients diverge away from any local minima present and towards 0 instead so the cost function w.r.t the input isn't minimised effectively as demonstrated in figure 1.

).

] .

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) 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

[BN is a process similar to whitening which when impleented on a deep neural network architecture allows the use of higher leaaring weights while also reducing the reliance on parameter initialization when training a deep neural network for optimal perfromance (?). It achieves this by normalizing the activations used within each leayer, effectively whiteining the inputs of each layer. This means the ditributions of layer inputs remain fixed hence allowing the network to converge faster since the effects of ICS are reduced. For a model trained using mini-batch SGD, the process can be described by the following equations:

$$\mu = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)} \tag{3}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}^{(i)}-\mu)^2 \tag{4}$$

$$\mathbf{x}^{(i)}_{new} = \frac{\mathbf{x}^{(i)}-\mu}{\sqrt{\sigma^2+\epsilon}} \tag{5}$$

$$\mathbf{z}^{(i)} = \gamma\mathbf{x}^{(i)}_{new} + \beta \tag{6}$$

Equatoins (3) and (4) describe the mean and varaince across a mini-batch of size m wehre $\mathbf{x}^{(i)}$ represents each training instance within the mini-batch. These mini-batach statistics are used in equation (5) to compute the zero-centered normalized activations where ϵ is a small constant used to prevent zero division. The reuslts obtained in (5) are then used in (6) to scale and shift these inputs linearly such that it allows any activation to be manipulated to non-linearities fittingly. γ and β in (6) are learned parameters that are calculated during backpropagation. All of the new activations obtained from equation (6) have fixed means and varainces hence the introduction of these norrmalized inputs should accelerate training and convergence of the network.

During training and test-time BN is able to converge significantly faster towards a recorded baseline accuracy as highlighted in (Ioffe & Szegedy, 2015). BN achieved 72.2% validation accuracy in less than half the training steps compared to Inception. Furthermore, since BN allows for higher learning rates, this led to reuslts beating Inception by over 11x when further parameters were tuned accoding to 4.2.1 in (Ioffe & Szegedy, 2015). It should be noted however that the overal time to train across a fixed number of

epochs wull increase compared to a baseline not using BN. This is due to each normalized layer reuqiring 4 addtional parameters to be learned for each deature($\mu, \sigma, \beta, \gamma$), thus the computatinal complexity of the model is greater.

$$E[x] \leftarrow E_{\mathfrak{B}}[\mu_{\mathfrak{B}}] \tag{7}$$

$$Var[x] \leftarrow \frac{m}{m-1} E_{\mathfrak{B}}[\sigma^2_{\mathfrak{B}}] \tag{8}$$

The averages for mean and varaince across multiple mini-batches $\mathfrak{B}$ are taken to estimate the population parameters which are once again used in equation (5) but with the population statistics substiuted instead. The final BN linear transofmraiton fomrula becomes:

$$\mathbf{z} = \frac{\gamma}{\sqrt{Var[x] + \epsilon}}.\mathbf{x} + \left(\beta - \frac{\gamma E[x]}{\sqrt{Var[x] + \epsilon}}\right) \tag{9}$$

where $z = BN_{\gamma,\beta}(x)$

Batch Normalization addresses the vanishing gradient problem by enusring activations don't map to saturation points when passed thorugh it's corresponsing function. It achieves this by using the learned γ and β values to scale activations before being passed through non-linearities. Due to the normalization across all activations, the model is no longer sensitive to small parameter changes which would usually push the activations towards saturation points and cause the vanishing gradient problem. Batch Normalization keeps mappings within areas of non-saturation to allow for deeper networks to be trained successfully.

] .

4.2. Residual connections

[Residual connections descirbe a technique used to improve the performance of deep neural networks which contain many layers. It does this by hypothesizing that it is easier to optimise a residual mapping across stacked latyers compared to the original mapping (**He et al., 2016**). This means that if an identity mapping was found to be optimal across layers, it would be easy to push the resiual mapping to zero rather than trying to fit an identity mapping through a stack of non-linear layers (**He et al., 2016**). The technique used to obtain the deisred output of the neural network using the residual mapping hypothesis can be decribed by the following equation:

$$H[x] = F(x) + x \tag{10}$$

$H(x)$ describes the desired underlying mapping of the model which is obtained by summing residuals with the output of the original mapping thorugh the stacked non-linear layers $F(x)$. The reiduals x are connected via "shortcut connections" from the input layer directly to the output of the stacked layers Fx hence skipping

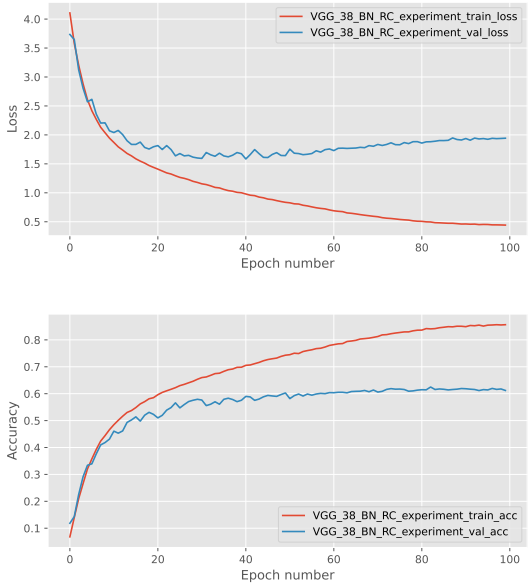


Figure 4. Training curves for VGG 38 using BN+RC

computation through the stacked layers. Since shortcut connections are simply identity mappings, the use of this technique adds neither parameter nor computational complexity during training and testing thus can be implemented efficiently using common libraries (**He et al., 2016**).

RCs aid in solving the vanishing gradient problem by providing layers with extra information in the form of residuals/identity mappings. This ensures the original input data is constantly propogated thorughout the deep network which enhances its gradient flow. Take the example of samll values less than 1 being multiplied across stacked layers during a forward pass. Naturally, these values would converge towards zero. However, since residuals from skip connections are summed at the outputs, values are prevented from converging towards zero quickly. This allows the gradient to be propogated further through the network without being a victim to the vansihing gradient problem.

] .

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 strategies (cropping, horizontal flipping) to improve the generalization of the model.

With the goal of understanding whether BN or skip connec-

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	1.65	53.61	2.01	46.79
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	0.44	85.64	1.87	62.00

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.

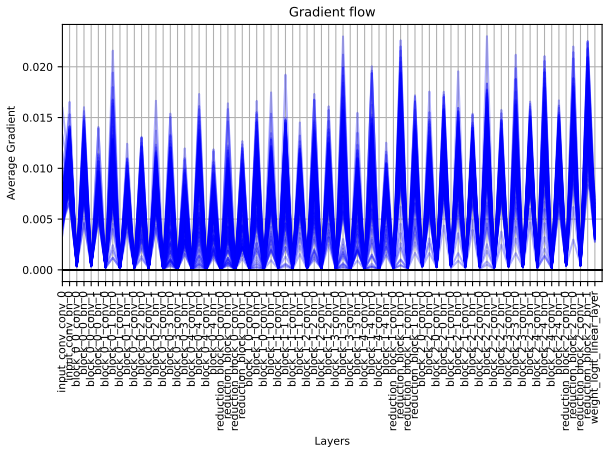


Figure 5. Gradient Flow on VGG 38 using BN+RC

tions 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

[From table 1, we can deduce that using Batch Normalization alone on the VGG 38 model served well as a baseline model for solving the vanishing gradient problem. We see this based on training and validation accuracies for both models using BN having ranges between 52.28%-53.61% and 46.72-46.79% respectively. That is a significant improvement from the model plagued by the vanishing gradient problem which had <1% validation accuracy. However, it can also be inferred that using batch normalization alone doesn't yield satisfactory results for learning features as the validation accuracy for both models stil remain <50%. It is clear that using BN helps reduce the training and testing error caused by the vanishing gradient but doesn't contribute much alone to imporving the learning of the model.

On the contrary, it can be seen that using residual connections alone yields a substantial gain in both training and validation accuracy. Results show there is a 6-8% improvement in training and testing validation perfromance when RC is utilised over BN. Furthermore, there is a clear drop in training and test error over the BN model which shows signs of RC being a superior method to impeove a deep neural network's performance.

Taking the model a step further by utilising both BN and RC across all layers yields the most optimal results. When using an increased learning rate of 10^{-2}, the combined model yields a validation accuracy of 62% which is a stageering 15% better than the BN-only models. There is strong evidence to suggest using a combination of both techniques yields a model capable of capturing more complex features. Figure 5 highlights what the gradient flow of a network learning effectively should look like versus what a poorly learning network looks like as shown in figure 3.

Although the BN+RC model yielded the best performance, it also highlighted a new issue of overfitting which is seemingly occuring. Figures 4 and 5 portray an increasing generalization gap as the model learns across 100 epochs. At around epoch 40, we can observe the depreciation of validation loss becoming stagnant whilst the training loss continues to decrease significantly. By epoch 100 the loss is at 1.87 versus a loss of 1.58 after epoch 40.

To improve both the performance of the model and to avoid overfitting, techniques covered in the previous paper can be implemented. Section 4.2.1 in (Ioffe & Szegedy, 2015) highlights key parameter changes made which drastically improved validation reuslts. From the reuslts published in the paper it had been shwon

that increasing the learning rate by 30x more than the baseline yielded peak results. Therefore in a future experiment, this course of action would be taken as well as introducing l2 regularization and tuning its parameters to control the overfitting that may be intoroduced with the use of higher learning rates. Experiments consisting of higher learning rates using BN+RC with and without L1/L2 regularization should be conducted and compared. These experiments would prove ideal as it would give an indication as to how much BN allows for the learning rate to be increased without overftting occurring, and if these effects can be mitigated with regularization whilst maintaining very high learning rates.

In addition, experimenting using varying amounts of skip connections would allow the model's hyperparameters to be finetuned further to obtain the ideal amount of skip connections required for optimal validation performance and ideal generalization.

] .

7. Conclusion

[From the results obrained via the various experiments conducted,it can be observed that the use of BN and RC cohesively proves to be the optimal solution when mitigating against the vanishing gradient problem whilst optimising model perfomance. Using RC alone does yield better performance but ideally it should be acompanied with BN. This allows higher learning rates to be used to further improve the model. Using BN alone doesn't yield very strong model performance however it does alleviate the vanishing gradient problem hence allowing deeper networks to be trained.

As detailed earlier in the paper, suggestions from section 4.2.1 of (Ioffe & Szegedy, 2015) should be experimented with. Specifically, the combination of using higher learning rates wihlst introducing L2 regularization should be trialled. This may help answer the question of determining the upper limit of how quickly we can train a model witout it becoming overfit. Furthermore, varying the amount of skip connections and layers skipped can allow us to determine the ideal hyperparameters required for this model to perform optimally.] .

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.