MLP Coursework 2

s2020491

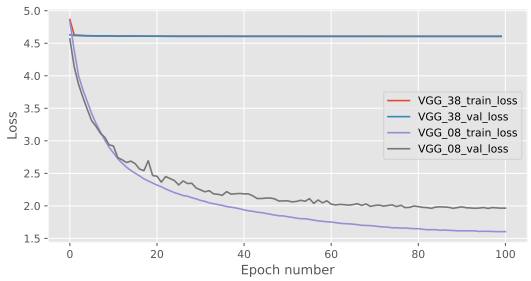
Abstract

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.

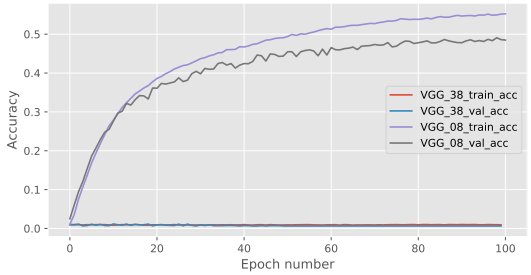
1. Introduction

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

This report focuses on diagnosing the VGP occurring in the VGG38 model[1] and addressing it by implementing two standard solutions. In particular, we first study a "broken" network in terms of its gradient flow, L1 norm of gradients with respect to its weights for each layer and contrast it to ones in the healthy and shallower VGG08 to pinpoint the problem. Next, we review two standard solutions for this problem, batch normalization (BN) (Ioffe & Szegedy, 2015) and residual connections (RC) (He et al., 2016a) in detail and discuss how they can address the gradient problem. We first incorporate batch normalization (denoted as VGG38+BN), residual connections (denoted as VGG38+RC), and their combination (denoted as VGG38+BN+RC) to the given VGG38 architecture. We train the resulting three configurations, and VGG08 and VGG38 models on CIFAR100 (pronounced as 'see far 100') dataset and present the results. The results show that though separate use of BN and RC does mitigate the vanishing/exploding gradient problem, therefore enabling effective training of the VGG38 model, the best results are obtained by combining both BN and RC.



(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

2. Identifying training problems of a deep CNN

[ ]

Concretely, training deep neural networks typically involves

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

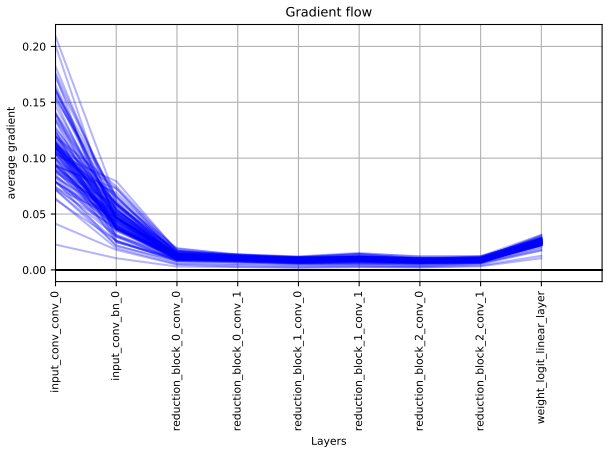


Figure 2. Gradient flow on VGG08

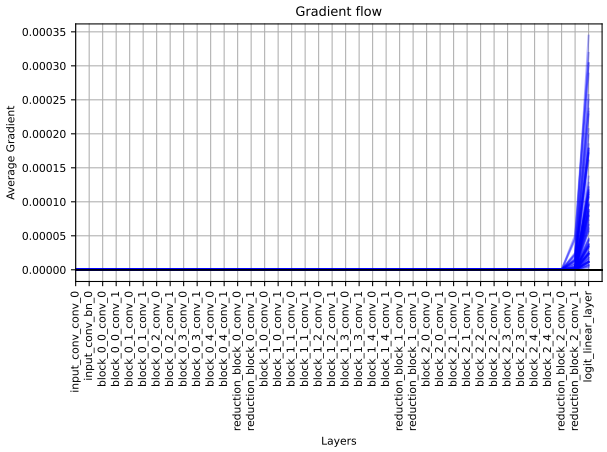


Figure 3. Gradient Flow on VGG38

three steps: forward pass, backward pass (or backpropagation algorithm (Rumelhart et al., 1986)) and weight update. The first step involves passing the input $\boldsymbol{x}^{(0)}$ to the network and producing the network prediction and also the error value. In detail, each layer takes in the output of the previous layer and applies a non-linear transformation:

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

where (l) denotes the l-th layer in L layer deep network, $f^{(l)}(\cdot, W^{(l)})$ is a non-linear transformation for layer l, and $W^{(l)}$ are the weights of layer l. For instance, $f^{(l)}$ is typically a convolution operation followed by an activation function in convolutional neural networks. The second step involves the backpropagation algorithm, where we calculate the gradient of an error function E (*e.g.* cross-entropy) for each layer's weight as follows:

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

This step includes consecutive tensor multiplications between multiple partial derivative terms. The final step involves updating model weights by using the computed $\frac{\partial E}{\partial W^{(l)}}$ with an update rule. The exact update rule depends on the optimizer.

A notorious problem for training deep neural networks is the vanishing/exploding gradient problem (Bengio et al., 1993) that typically occurs in the backpropagation step when some of partial gradient terms in Eq. 2 includes values larger or smaller than 1. In this case, due to the multiple consecutive multiplications, the gradients *w.r.t.* weights can get exponentially very small (close to 0) or very large (close to infinity) and prevent effective learning of network weights.

Figures 2 and 3 depict the gradient flows through VGG architectures (Simonyan & Zisserman, 2014) with 8 and 38 layers respectively, trained and evaluated for a total of 100 epochs on the CIFAR100 dataset. **[VGG38 suffers from the vanishing gradient problem as shown by the gradients dropping off to zero after only three layers. VGG08 on the other hand does not appear to suffer from the vanishing gradient problem as at no points do the gradients settle at zero, and across the duration of the model they are orders of magnitude higher than the tiny gradients in VGG38.**

The consequences of this problem are shown in Figure 1, where the first few epochs mildly decrease loss, followed by an immediate plateau in loss and accuracy. As such, the model can only slightly attempt to train through the first few layers, after which it is unable to, which gives the tiny final accuracy value of 00.01 which is even worse than if the model just guessed at random for each case.] .

3. Background Literature

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

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

It is hence beneficial to embed the normalization of training data into the network architecture after work from LeCun *et al.* showed that training converges faster with this addition (LeCun et al., 2012). Through standardizing the inputs to each layer, we take a step towards achieving the fixed distributions of inputs that remove the ill effects of ICS.

Ioffe and Szegedy demonstrate the effectiveness of their technique through training an ensemble of BN networks which achieve an accuracy on the ImageNet classification task exceeding that of humans in 14 times fewer training steps than the state-of-the-art of the time. It should be noted, however, that the exact reason for BN's effectiveness is still not completely understood and it is an open research question (Santurkar et al., 2018).

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

[In the work of (He et al., 2016a), results show that the deeper plain 56-layer network performs worse than the shallower plain 20-layer network for each epoch on the CIFAR-10 dataset. This was the case for both the training and the test errors. Deeper networks can be more prone to overfitting due to an increased network capacity resulting in the memorising of the training data. This is indicated by an increasing training accuracy, but a decreasing validation accuracy, or in other words an increasing generalisation gap. A smaller network capacity may struggle to fit around the training set to a suitable degree in the first place, preventing a high training accuracy from ever being reached. These indicators are not present in our Figure 1 or Figure 1 in (He et al., 2016a) as the deeper network begins to somewhat plateau its training error far above the shallower network's error. To be able to train properly, the network needs to be able to optimise properly. This cannot be accomplished on a plain deep network due to the vanishing gradient problem preventing learning from happening later on in backpropagation. Alternative factors that prevent the larger network from achieving a high training accuracy could be an inappropriate learning rate or a poor experimental setup. This could include not employing stochastic/mini-batch gradient descent, an appropriate optimiser or scheduler, or poor initialisation techniques. All of these factors could lead to the network getting caught in suboptimal regions or even local minima.] .

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

4. Solution overview

4.1. Batch normalization

BN has been a standard component in the state-of-the-art convolutional neural networks (He et al., 2016a; Huang et al., 2017). Concretely, BN is a layer transformation that is performed to whiten the activations originating from each layer. As computing full dataset statistics at each training iteration would be computationally expensive, BN computes batch statistics to approximate them. Given a minibatch of B training samples and their feature maps $X = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^B)$ at an arbitrary layer where $X \in \mathbb{R}^{B\times H\times W\times C}$, H, W are the height, width of the feature map and C is the number of channels, the batch normalization first computes the following statistics:

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

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

where c, i, j denote the index values for y, x and channel coordinates of feature maps, and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the mean and variance of the batch.

BN applies the following operation on each feature map in batch B for every c, i, j:

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

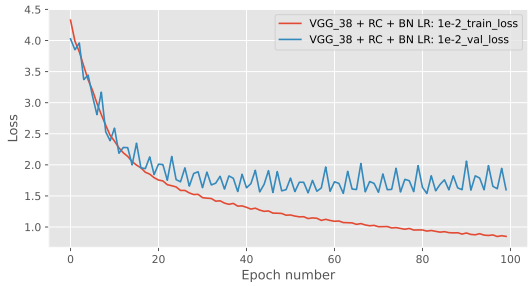
where $\gamma \in \mathbb{R}^C$ and $\beta \in \mathbb{R}^C$ are learnable parameters and ϵ is a small constant introduced to ensure numerical stability.

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.

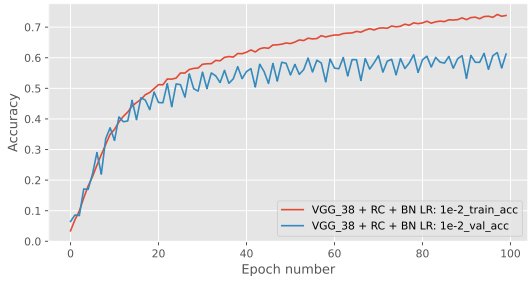
Recent work has shown that BatchNorm has a more fundamental benefit of smoothing the optimization landscape during training (Santurkar et al., 2018) thus enhancing the predictive power of gradients as our guide to the global minimum. Furthermore, a smoother optimization landscape should additionally 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., 2016a) to tackle the



(a) Cross entropy error per epoch



(b) Classification accuracy per epoch

Figure 4. Training curves for VGG38 with residual connections, batch normalisation and a learning rate of 1e-2, in terms of (a) cross-entropy error and (b) classification accuracy

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

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

$$\frac{\partial \boldsymbol{x}^{(b)}}{\partial \boldsymbol{x}^{(b-1)}} = \mathbb{1} + \frac{\partial F(\boldsymbol{x}^{(b-1)})}{\partial \boldsymbol{x}^{(b-1)}} \tag{6}$$

where $\boldsymbol{x}^{(b-1)} \in \mathbb{R}^{C\times H\times W}$ and $\mathbb{1}$ is a $\mathbb{R}^{C\times H\times W}$-dimensional tensor with entries 1 where C, H and W denote the number of feature maps, its height and width respectively. Importantly, $\mathbb{1}$ prevents the zero gradient flow.

5. Experiment Setup

[ ]

[ ]

[ ]

We conduct our experiment on the CIFAR100 dataset (Krizhevsky et al., 2009), which consists of 60,000 32x32

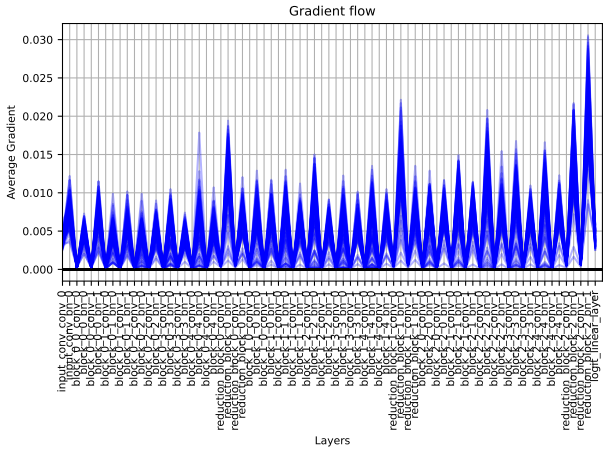


Figure 5. Gradient Flow on VGG38 with batch normalisation, residual connections and a learning rate of 1e-2

colour images from 100 different classes. The number of samples per class is balanced, and the samples are split into training, validation, and test set while maintaining balanced class proportions. In total, there are 47,500; 2,500; and 10,000 instances in the training, validation, and test set, respectively. Moreover, we apply data augmentation strategies (cropping, horizontal flipping) to improve the generalization of the model.

With the goal of understanding whether BN or skip connections help fighting vanishing gradients, we first test these methods independently, before combining them in an attempt to fully exploit the depth of the VGG38 model.

All experiments are conducted using the Adam optimizer with the default learning rate (1e-3) – unless otherwise specified, cosine annealing and a batch size of 100 for 100 epochs. Additionally, training images are augmented with random cropping and horizontal flipping. Note that we do not use data augmentation at test time. These hyperparameters along with the augmentation strategy are used to produce the results shown in Fig. 1.

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

5.1. Residual Connections to Downsampling Layers

[Given downsampling layers reduce the dimensions of the data being passed on, to appropriately add residual connections we would need to reduce the input data as well to match the dimensions required before perform-

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	1.74	51.49	1.95	46.36
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	0.849	73.82	1.60	61.20

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 the addition. In our models, downsampling layers reduce the spatial dimensions (height and width) of feature maps by a factor of two through average pooling.

One approach is to use a 1x1 convolution in the skip path with a stride of 2. This has the effect of halving both the height and the width, ensuring both the spatial dimensions and the number of channels are aligned, allowing the addition of the input and output. This method is widely used in architectures like ResNet (**He et al., 2016a**).

Alternatively, a simpler approach is to use average pooling in the skip path to reduce the spatial dimensions. This is computationally cheaper than using the convolution, making it suitable for lightweight models. However, the pooling operation can lose fine-grained information from the input, potentially impacting the network's ability to learn intricate patterns.

The convolutional approach, while slightly more computationally expensive, allows for a learnable transformation which enables the network to preserve and adapt critical features. Therefore, while pooling offers simplicity and efficiency, 1x1 convolutions are more versatile and much more able to retain more meaningful representations in the residual path.] .

6. Results and Discussion

[Both batch normalisation and residual connections individually seemed able to fix the vanishing gradient problem on our VGG38 implementation as expected. This was shown by the massively increased validation accuracy from 0.01% in the original VGG38 with 46.36% and 52.32% for batch normalisation and residual connection respectively. It was further demonstrated by Figure **4** where the loss significantly decreases and the accuracy significantly increases across epochs, indicating the model can learn. We can also see when comparing Figures **3** and **5** that the gradient no longer drops to and remains at 0, implying each layer is able to train to minimise the loss function, as opposed to the original model where only the first few layers were affected. While the gradients in Figure **5** never reach quite as high as in the first couple of layers, they remain at a consistently high level throughout. Figure **4** shows signs of overfitting as one would expect in a model of such high capacity, as indicated by an increasing generalisation gap starting around epochs 12-16. Despite the highest training accuracy only reaching 73.82%, the accuracy curve appears to still be increasing which implies the network architecture has the capacity to reach even higher values given more epochs. Similarly, the validation accuracy appears to still be increasing which could imply that simply increasing the number of epochs would give improved results. It is not surprising that the accuracies had not converged within 100 epochs due to the depth of the model. The work in (**He et al., 2016b**) yields an accuracy on the CIFAR-100 dataset of 77.29 which shows there to be plenty of room for improvement.
The VGG38 RC experiment yielded a significantly better result than the VGG38 BN experiment, suggesting superior efficacy. The inclusion of both solutions has improved the results on the VGG38 network, albeit only slightly more than with the test with RC on its own, which suggests they compound well. We also partially investigated how changing the learning rate would affect performance. Increasing the learning rate from 1e-3 to 1e-2 had a significant impact on final accuracies when applying both RC and BN, producing our best-performing model with a validation accuracy of 61.2%. This was a significant increase on the same model architecture trained with a learning rate of 1e-3 which yielded a validation accuracy of 53.76%, and on the VGG08 model which yielded only 46.84%. The impact of the learning rate change was only recognisable with this setup, however, as the same change made on models that only implemented batch normalisation resulted in a validation accuracy improvement of 0.36%. An alternative to increasing the number of epochs could be to increase the learning rate to speed up convergence, thus yielding improved performance. However, the results we have collected are not sufficient to draw any reliable generalised conclusion about the comparative efficacies of the methods.
I would wish to run a test with just residual connections and a learning rate of 1e-2 to explore in greater detail if residual connections are a more impactful modification on performance than batch normalisation for this type of task. I also wish to perform VGG38 with both RC and BN with either a higher learning rate or an increased

number of epochs to test the upper limit of training convergence. Beyond this, it would be suitable to implement regularisation as well to prevent the overfitting that our results are now demonstrating, and hopefully increase the validation accuracy as a result. Finally, I wish to test these architecture permutations on a different type of data such as sensory. The aim of this is to try and generalise if RC is consistently better than BN, or if this seems to be a preference related to image data.] .

7. Conclusion

[This report presented the vanishing gradient problem in the context of CNN models, demonstrated on the CIFAR 100 dataset. We then evaluated the impact of implementing two potential changes to prevent this problem: Residual Connections and Batch Normalisation. Both approaches were able to solve the vanishing gradient problem which allowed the increased network capacity of VGG38 to give far better results than the much shallower VGG08 model. However, even the most successful combination of model parameters was still only able to give a validation accuracy of 61.20% which it was suggested could be improved upon with a larger learning rate or increased number of epochs as neither the training or validation accuracies appeared to have converged.

The number of blocks was chosen arbitrarily and as such there is likely a lot of room for determining the most efficient network depth without loss of accuracy. Initially, this could be accomplished by training and testing an array of models of varying depths with a sufficient number of epochs and regularisation, whether that be layers per block or the number of blocks. From there we could more easily determine a preferred trade-off between complexity and accuracy. An alternative could be to determine across the training set if there exists channels or layers in particular that are barely used in the classification process, which could allow pruning for a more lightweight model that doesn't compromise performance.

An extension beyond this work could include extending the residual networks we have created with transformer architecture as done in (**Carion et al., 2020**). This approach improves upon the handling of non-local features due to a transformer's ability to attend to each of them without downsampling which comes at the cost of spatial resolution. As such, we could expect to see an improvement beyond the reaches of our current architecture.].

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.

Carion, Nicolas, Massa, Francisco, Synnaeve, Gabriel, Usunier, Nicolas, Kirillov, Alexander, and Zagoruyko, Sergey. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pp. 213–229. Springer, 2020.

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, 2016a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pp. 630–645. Springer, 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.