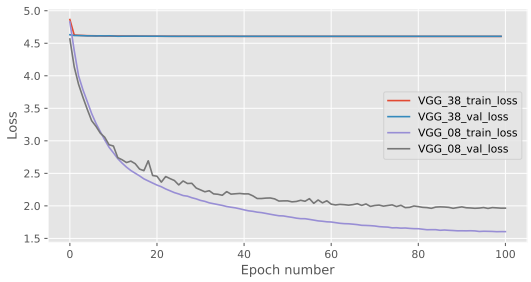
MLP Coursework 2

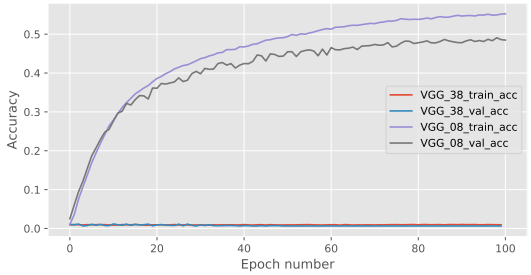
S25697578

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) 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

1. Introduction

Despite the remarkable progress of modern convolutional neural networks (CNNs) 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[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., 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 CIFAR100 (pronounced as 'see far 100') dataset and present the results. The results show that though separate use of BN and RC does mitigate the vanishing/exploding gradient problem, therefore enabling effective training of the VGG38 model, the best results are obtained by combining both BN and RC.

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

2. Identifying training problems of a deep CNN

[Question Figure 3]

Concretely, training deep neural networks typically involves

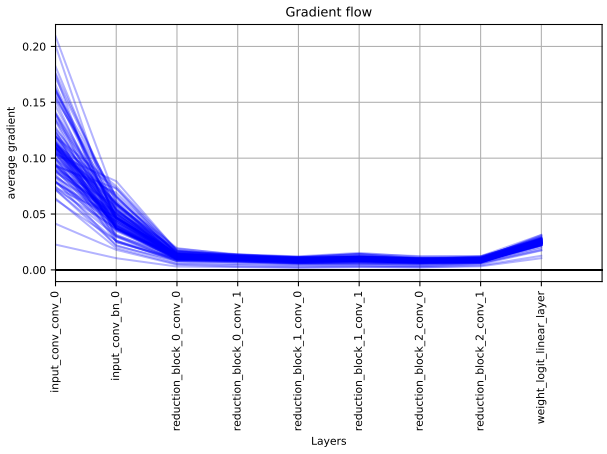


Figure 2. Gradient flow on VGG08

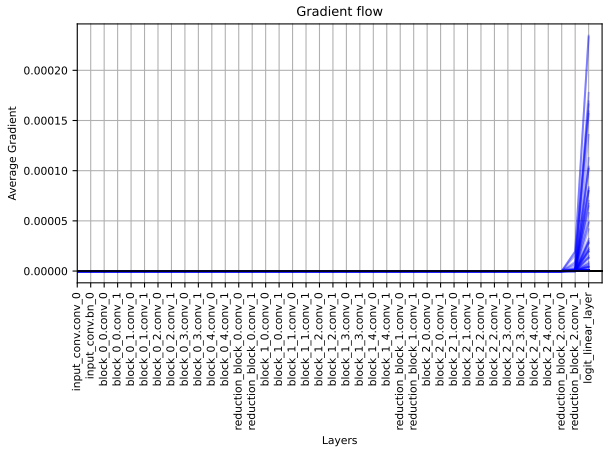


Figure 3. Gradient Flow on VGG38

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. **[From Figure 2 and 3, we can see that there is a healthy amount of gradient flow across the different layers of VGG 08 model, but not the VGG 38 model. This indicates that the gradient is not flowing properly through the layers of the VGG 38 model pointing that it suffers from a vanishing gradient problem which is hampering the training of the model. This is further supported by Figure 1 (a) and (b) which show that the loss of the VGG 38 model remains high while the accuracy being very low on both the training and the validation datasets.]** .

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.

[A complex model like VGG38 has a lot of network capacity due to its deeper architecture. But a complex model like this is more likely to encounter the vanishing gradients problem, hindering effective weight update in its earlier layers (Figure 3). This is too be observed because VGG38 has an exteremly deep architecture and is made to handle more complex tasks and larger datasets. The image size of the CIFAR100 class is only 32×32 with 100 classes which is very less for VGG38. In contrast, a shallower model with relatively less network capacity (VGG08), exhibits normal gradient flow and is able to use its architecture and depth more efficiently to learn the data and train a relatively better model. Some other factors that may have contributed to this difference include the model architecture, optimisation techniques used and the quality of the dataset.] .

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., 2016; 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., 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

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

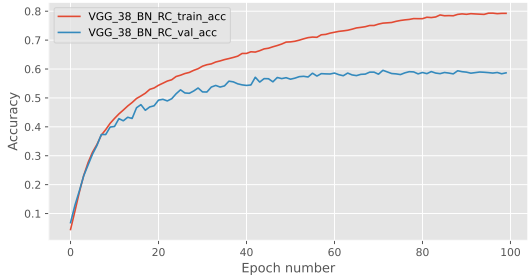


Figure 4. Training curves for VGG38 with Batch Normalisation and Residual Connections in terms of classification accuracy

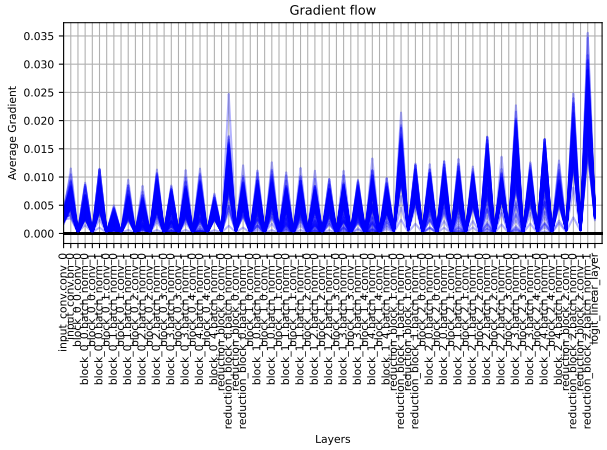


Figure 5. Gradient Flow on VGG38 with Batch Normalisation and Residual Connections

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

[Question Figure 4]

[Question Figure 5]

[Question Table 1:

***VGG38 BN (LR 1e-3)* and *VGG38 BN + RC (LR 1e-2)*.]**

We conduct our experiment on the CIFAR100 dataset (Krizhevsky et al., 2009), which consists of 60,000 32x32 colour images from 100 different classes. The number of samples per class is balanced, and the samples are split into training, validation, and test set while maintaining balanced class proportions. In total, there are 47,500; 2,500; and 10,000 instances in the training, validation, and test set, respectively. Moreover, we apply data augmentation strategies (cropping, horizontal flipping) to improve the generalization of the model.

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

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

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

[Residual connections are difficult to implement in downsampling layers because of the mismatch in the dimension from the residual to the point where it is being added back to the network. We would have to alter the model architecture in order to incorporate this ability in our model. This residual connection would serve as a shortcut for the gradients to directly flow through, skipping layers and preserving the gradient information helping in better training of deep models. There are 2 primary ways to implement this - Direct Residual Connections & Bottleneck Residual Connections (He et al., 2016):

- **Direct Residual Connections: In this method, the input gradients are directly added to the output of the downsampled layer. This method is very intuitive and relatively easy to implement in the model architecture. If we were to look at the downsides of this method, it still carries the risk of attracting the phenomenon of vanishing gradients in very deep models. It might also be not very efficient for models with extensive depth.**

- **Bottleneck Residual Connections: This method utilises 1×1 convolutions to reduce and expand the number of features in the model enabling healthier gradient flow. This also helps in reducing the number of parameters effectively reducing the computational cost of the model. Although it is better at tackling the issue of vanishing gradients in very deep models, it is more complex than Direct Residual Connections and adds extra 1×1 convolutions to the model architecture.**

] .

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	53.33	1.86	48.60
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	0.96	71.01	1.63	58.96

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.

6. Results and Discussion

[Variety of experiments were performed on the VGG38 model implementing a plethora of techniques to combat the vanishing gradient problem and then check its performance on the CIFAR100 dataset. A baseline VGG08 model was used a reference point which did not suffer from a vanishing gradient problem like the baseline VGG38 model did.

We introduced batch normalisation and residual connections in the VGG38 model architecture to combat the problem of vanishing gradients. Batch Normalisation prevents the weights from getting too large or small ensuring a healthy flow of the gradient through the layers. It is to be noted that the addition of Batch Normalisation adds approximately *3,000* parameters to the model increasing the computational cost to train it. Direct Residual Connections (Section 5.1) does not add any parameters to the model. It ensures the preservation of gradient in training of very deep models.

The introduction of these modifications instantly fixes the problem of vanishing gradient and the model starts to learn some features from the data. Keeping the learning rate constant, the end results are marginally better than that of the baseline VGG08 model but the loss is still relatively high and there is still room for improvement in the accuracy of the model (Table 1).

We then combined the two methods to see if they boost the model performance. There was little improvement when compared to the model with residual connections indicating that other parameters of the model have to be changed to get better results. The learning rate (LR) was increased to *0.01* from *0.001*. This was first changed in the model with only batch normalisation to study its effects on the model. This variation ended up performing worse than the previous models (Table 1).

This new learning rate was then applied to the VGG38 model with both Batch Normalisation and Residual Connections. This significantly improved the model's performance on the train set but not that much test set. This combination of large learning rate with BN and RC has ensured the healthy flow of gradient, but due to its huge network capacity and small dataset for its size, the model is still overfitting on the data.

Taking this experiment further, we can try modifying the LR in the VGG38 model with just RC and see how the relatively big jumps in gradients are influenced by shortcuts in the different layers of the CNN.] .

7. Conclusion

[Question 5 - Briefly draw your conclusions based on the results from the previous sections (what are the take-away messages?) and conclude your report with a recommendation for future work.

Good recommendations for future work also draw on the broader literature (the papers already referenced are good starting points). Great recommendations for future work are not just incremental (an example of an incremental suggestion would be: "we could also train with different learning rates") but instead also identify meaningful questions or, in other words, questions with answers that might be somewhat more generally applicable.

For example, (Huang et al., 2017) end with

> **"Because of their compact internal representations and reduced feature redundancy, DenseNets may be good feature extractors for various computer vision tasks that build on convolutional features, e.g., [4,5]."**

while (Bengio et al., 1993) state in their conclusions that

> **"There remains theoretical questions to be considered, such as whether the problem with simple gradient descent discussed in this paper would be observed with chaotic attractors that are not hyperbolic."**

The length of this question description is indicative of the average length of a conclusion section] .

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.