Finding the Right Teacher for a Difficult Student

Daniel Whettam

Master of Science

Data Science

School of Informatics

University of Edinburgh

2020

Abstract

Network distillation is an approach to effectively training neural networks when constrained by the size of the network. Network distillation (Ba and Caruana, 2014; Hinton et al., 2015; Zagoruyko and Komodakis, 2016a) uses a large teacher network to guide the learnt representation of the network of interest, the student.

This project is an investigation into how to best determine an effective teacher for a given student when using network distillation, with a particular focus on cases where the student is of a non-standard architecture, such as one generated through neural architecture search. Specifically, we use Differentiable Architecture Search (DARTS) (Liu et al., 2019) to generate our non-standard architectures. We explore two key strands within the general objective of determining an effective teacher network for the student. Firstly, we explore different teacher-student pairings, comparing models generated through neural architecture search with readily available, pre-trained models such as a WideResNet (Zagoruyko and Komodakis, 2016b) and DenseNet (Huang et al., 2017). We find that both the WideResNet and DenseNet models pair well together, and both models struggle to teach a student generated through DARTS effectively. We conclude that, in general, the best teacher for any student is the one that is most similar to the student. We evaluate this finding through an in-depth discussion and analysis of the results and visualisation of the attention maps for a variety of teacher-student pairings.

Secondly, we propose a novel approach to developing a teacher network from the student that uses Fisher information to determine which cells in the network to grow. We grow the cells by reducing the number of groups in the grouped convolutions that occur within that cell, effectively increasing the number of channels in the cell. We call this approach Fisher expansion and demonstrate that it gives improved results over a student trained independently, a student trained using a teacher developed through a depth-wise scaling, as well as a teacher developed through a uniform growth of the cells. We use these results to confirm our hypothesis that what is most important when choosing a teacher-student pairing is the similarity between the two architectures and that a channel-wise scaling of a model creates a much more effective teacher architecture than a depth-wise scaling.

Acknowledgements

I'd like to thank my supervisors, Amos Storkey and Elliot Crowley, for their help and guidance throughout this project. Amos' advice early on in the project, and feedback in more recent weeks was fundamental in the development of this project. I'd particularly like to thank Elliot Crowley, who gave me invaluable advice, regularly offering thoughtful suggestions for problems that I would still be trying to resolve today. I'd also like to thank the entirety of the BayesWatch group, who regularly offered help with debugging, and provided a constant feed of interesting papers and discussion that was extremely useful when writing this thesis.

I'd also like to thank my parents, Andrew and Rebecca Whettam, for their continual support during my time in Edinburgh, as well as all-important help with proofreading this project.

Finally, I'd like to thank Ana Prelici for her continual love and support.

Table of Contents

Chapter 1

Introduction

The use of neural networks has become widely established within the machine learning community since the introduction of AlexNet (Krizhevsky et al., 2012), developing into a sub-field known as deep learning. Deep learning attempts to approximate complex functions through the stacking of many layers of non-linear functions. These functions consist of weights which are updated through an optimisation procedure that minimises an error function. Through the error-function-minimisation process, a neural network can approximate the function that is being learnt. This general procedure accounts for a large amount of the successes within machine learning. Work (Simonyan and Zisserman, 2015; Szegedy et al., 2015) with convolutional neural networks (LeCun et al., 1998) has demonstrated that increasing the depth of neural networks offers significant performance gains; this follows a natural intuition that deeper networks allow for more complex functions to be learnt as a result of stacking increasing numbers of non-linearities. As a result of increasing the depth of a network, the number of parameters required can quickly become very large; this is not a problem on a large desktop computer; however, it can present issues when trying to run deep learning models on memory-restricted devices. This dissertation investigates ways to maintain the performance of deep neural networks while using fewer parameters. We experiment with network distillation (Ba and Caruana, 2014; Hinton et al., 2015), which uses a *teacher* network to teach a smaller *student*. Our work focuses on cases where the student is of some non-standard architecture, for example, networks generated through Differentiable Architecture Search (DARTS) (Liu et al., 2019). We consider a non-standard architecture to be any network that is not widely available, such that an effective teacher cannot be easily downloaded from an online repository. Specifically, we investigate teacher-student pairings with readily available models and

DARTS models and propose a modelling approach for developing a teacher model from the student.

1.1 Motivation

The performance of deep neural networks is largely limited by the number of learnable parameters in the network (LeCun et al., 2015), thus motivating the development of very deep networks (Shazeer et al., 2017). However, while deep networks can be very effective, they require large amounts of memory for storage, as well as huge amounts of compute power for training purposes. It has been widely demonstrated that many of the parameters within a neural network are largely redundant, suggesting that it may be possible to reduce the number of parameters required by many very deep networks, while at the same time maintaining performance. For example, Denil et al. (2013) are able to predict all of the weights of a model from only a small subset of the models' weights and demonstrate that many of the weights do not need to be learnt at all. Frankle and Carbin (2019) suggest that for any dense, randomly initialised feed-forward network, there exists a smaller sub-network that is capable of reaching comparable performance to the full network when trained independently, with a similar training budget. While this over-parameterisation is well established, it remains a challenge to train a network that does not have many redundant parameters.

One approach to tackling this problem of over-parameterisation is network distillation. In network distillation learnt representations from a larger (teacher) network are used to improve the performance of the typically smaller (student) network, although there are some exceptions (Furlanello et al., 2018). When trained on just the original labelled data, the student does not perform as well as the teacher. With distillation, the student network utilises the learnt representations of the teacher network to improve its performance. Distillation addresses the problems of over-parameterisation by maintaining the performance of the larger network in a model with significantly fewer parameters.

An important question within network distillation is how best to determine the architectures for these teacher and student networks. In cases where the student is a standard, off-the-shelf architecture, such as ResNet (He et al., 2016a), or DenseNet (Huang et al., 2017), a teacher of that same architecture can be used (Crowley et al., 2018). However, in cases where the student is particularly non-standard, it becomes difficult to determine an effective architecture for the teacher. If the student network is the product of some *Neural Architecture Search* (NAS) algorithm (Liu et al., 2019; Zoph et al., 2018), then determining a good teacher architecture becomes even more challenging. NAS is an automated approach to finding effective neural network architectures. This project tackles the question of how to determine a good teacher architecture for a given student network, with particular focus on cases where the student network may be difficult or non-standard (e.g. NAS). Through addressing this problem we hope to increase the applicability of network distillation in scenarios where very non-standard architectures are in use, hopefully increasing the potential of memory-constrained devices (e.g. mobile phones) to utilise neural network-based technologies.

1.2 Objectives

The overall aim of the project is to determine how best to select a teacher architecture given a student network obtained through some NAS approach. As a result, we hope to gain some understanding as to the most effective network pairings for distillation purposes. We will approach our aim through the following objectives:

1. Investigate how effectively off-the-shelf teacher architectures can perform at object recognition when using a non-standard student architecture.
2. Develop an effective approach to modelling a teacher architecture when given a student network.

Following these objectives, there are a few key considerations. Most likely, some modelling process is necessary to develop an effective teacher architecture. However, it is also possible that by using a deeper NAS model as the teacher, and a shallower model as the student, comparable results can be achieved. Additionally, it is also possible that expanding a student using Fisher information may not offer any increased gains in performance; this should be tested by a comparison between a Fisher expanded teacher, and a NAS model that is scaled up in a naive fashion.

1.3 Contributions

This work presents two key contributions. Firstly, we implement a DARTS model into a distillation code-base. Using this implementation, we then provide experimental results for distillation with a range of teacher-student combinations, including models derived through a DARTS NAS process on CIFAR-10 — a standard benchmark dataset for image recognition. These results show that DARTS students typically under-perform when using either WideResNets or DenseNets as a teacher network. We also show that when a DARTS model is used as a teacher, it provides competitive results, and offers the best results for a DARTS student without using any sort of teacher modelling process. From these empirical results, we offer intuition regarding the sorts of network architectures that pair well together for distillation, suggesting that similar architectures tend to outperform architectural pairings that are less alike.

Secondly, we introduce and implement a new algorithm, Fisher expansion, for scaling up a DARTS network using Fisher information to rank the cells in a DARTS model. Using this cell ranking, we increase the capacity of the most impactful cells by reducing the number of groups in any grouped-convolution operations that occur within the cell. While applied specifically to DARTS models, our approach to scaling up a network could be applied to many different architectures. We provide experimental results of distillation for architectures derived using our Fisher expansion algorithm. Our results show that teachers created by expanding a student using its Fisher information outperform a teacher obtained by scaling the depth of the student, as well as a teacher that is developed from a naive, uniform channel expansion of the student.

1.4 Outline

Following this introductory chapter, Chapter 2 will cover all necessary background knowledge, including a discussion of the relevant literature. Chapter 3 will give a detailed explanation of our method, explaining how we go about using Fisher information to expand a student model, as well detailing our approach for testing distillation with combinations of off-the-shelf networks and NAS-derived models. Chapter 4 lays out our experiments and results. Chapter 5 provides discussion and analysis of the results, and Chapter 6 concludes our findings.

Chapter 2

Background

2.1 Deep Learning

As mentioned in the previous chapter, since the introduction of AlexNet in 2012, deep learning has become an extremely popular and effective branch of machine learning. Within computer vision, the state of the art for many important and difficult tasks such as object detection (Li et al., 2019), semantic segmentation (Zhu et al., 2019), and image recognition (Tan and Le, 2019) use deep learning, primarily with convolutional neural networks (LeCun et al., 1998) (CNN). Two popular and widely used CNN architectures are DenseNet (Huang et al., 2017) and WideResNet (Zagoruyko and Komodakis, 2016b). These models are very standard within image recognition and will be used as both teachers and students within our network distillation experiments. Typically, deep learning architectures are developed through a manual, iterative process, whereby authors may suggest improvements upon the previous state-of-the-art. Alternatively, architectures can also be developed through neural architecture search, which automates the process of finding deep learning architectures using a variety of possible approaches, such as reinforcement learning, and evolutionary algorithms.

2.1.1 Convolutional Neural Networks

Convolutional neural networks are a type of neural network typically used for computer vision tasks (e.g. image recognition, semantic segmentation). However, CNNs also have some application in other fields, such as speech recognition (Collobert et al., 2016). Standard feed-forward neural networks learn weights for each input dimension at each layer. In contrast, CNNs use a fixed kernel of weights that slides over the input (typically an image) and compute the dot-product between the kernel and image as the kernel moves across the image. The output is passed through a non-linearity, and the process is repeated, resulting in a deep network. The network is then optimised through some method of gradient descent, most commonly stochastic gradient descent (SGD) (Robbins and Monro, 1951).

Instead of inputting a raw image into a CNN, the image is typically split into *channels* (C), usually corresponding to the red, green and blue channels of the input image. As the input propagates through the network, these channels become more abstract and may increase in number. The kernel of weights also consists of channels, one for each of the input channels at the current layer. The total number of parameters in each layer of this network is then given by the number of channels multiplied by the height and width of the kernel: $C * h * w$. It is often standard practice to split the input along the channel dimensions, performing a smaller convolution on each group, and then concatenating the results to create an output. This process of splitting the input is referred to as a *grouped convolution* (Figure 2.1).

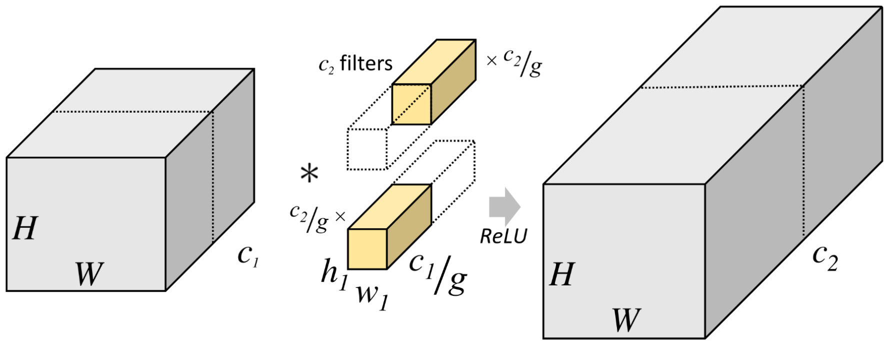


Figure 2.1: Figure from Ioannou (2017). In grouped convolutions, the input is split along the channel dimension, and the corresponding weights perform a convolution only on that subset of the input. After performing the convolutions, the outputs are then concatenated to create the final result.

By splitting the convolutions into groups, the number of parameters in the network is reduced. In the example shown in Figure 2.1, two groups are used. If the number of channels in each group is given by C_n/g, the total number of parameters used would be $C_1/g * C_2/g * h_1 * w_1 * g$, which is equivalent to $C_1 * C_2 * h_1 * w_1 * (1/g)$, the original number of parameters divided by the number of groups. Assigning the number of groups equal to the number of channels is referred to as a *depth-wise convolution*, and gives the fewest number of parameters. A single group is equivalent to a standard convolution and does not reduce the parameter count.

2.1.2 Wide Residual Networks

The WideResNet model (Zagoruyko and Komodakis, 2016b) (WRN) is an improvement upon an earlier model, known as ResNet (He et al., 2016a). As it has become increasingly clear that deep neural networks offer an increased representational power over their shallower counterparts (Bianchini and Scarselli, 2014), being able to effectively train these very deep networks has become an important research area. Training deep models is a difficult task due to issues of *degradation* (He et al., 2016a), where the model's accuracy begins to saturate, and then degrade, as depth increases. ResNet introduced a deep learning framework that can be easily trained, even when very deep. The key idea of this framework is the use of skip-connections. Instead of stacking layers in the normal fashion, a skip-connection is an identity mapping from previous layers to later layers (Figure 2.2). By incorporating these identity mappings, a deeper network should perform at least as well as its shallower counterpart; by using identity mappings for all later layers, the shallower network can be recreated. ResNets combine the identity mapping with the standard output through a simple summation, creating a single output to be passed into the next layer.

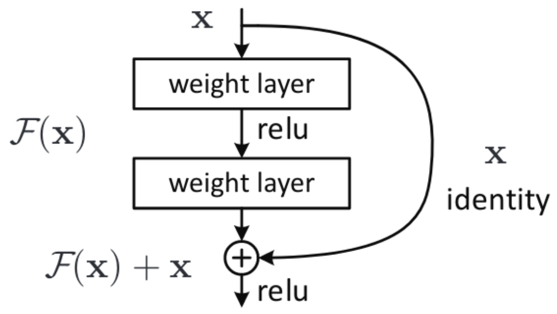

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.2: Figure from He et al. (2016a). The skip-connection and identity mapping introduced by ResNets. This increases the ability of the information in earlier layers to flow into the later layers of the network, allowing deeper networks to better utilise their increased representational power.

The key contribution of WideResNet is to increase the width of the convolutional layers in a residual network. The authors introduce a widening factor, k, that multiplies the number of features in each convolutional layer. Through increasing both depth and width, WideResNets offer an improvement in performance over the standard ResNet.

However, as the depth and width increase, so do the number of parameters, and the potential to overfit. In order to combat overfitting, a dropout layer (Srivastava et al., 2014) is added into each residual block, increasing performance on CIFAR-10 and CIFAR-100 by 0.11% and 0.4%, respectively.

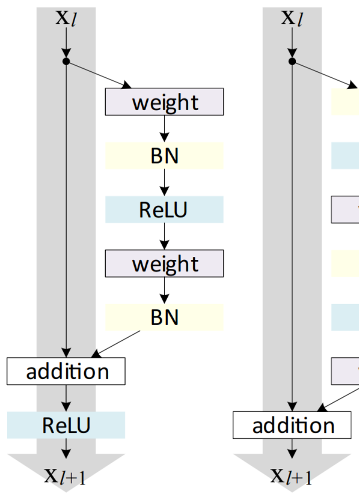


Figure 2.3: Figure from He et al. (2016b) *Left:* Post-activation - BatchNorm and ReLu are happening after the weight multiplication
Right: Pre-activation - BatchNorm and ReLu are happening prior to the weight multiplication

Additionally, WideResNets use *pre-activation* (He et al., 2016b), instead of the standard *post-activation*. Typically the activation function and batch normalisation (Ioffe and Szegedy, 2015) are applied after an affine transformation of each layers inputs, whereas pre-activation applies the activation function (e.g. ReLu) and BatchNorm before the affine transformation (Figure 2.3), and has been shown to give improved results (He et al., 2016b).

2.1.3 Dense Convolutional Networks

DenseNets provide a simple extension to the ResNet idea of skip-connections - connecting every layer in a network to every other layer via an identity mapping (Figure 2.4). In addition to connecting every layer, DenseNets concatenate the outputs, instead of summing them. This concatenation means the l^{th} layer will have l inputs, corresponding to the feature maps for all of the previous layers. Similar to WideResNets, DenseNets use a parameter, k, to refer to the models' growth rate. The growth rate determines how the number of feature-maps grows with each layer of the network.

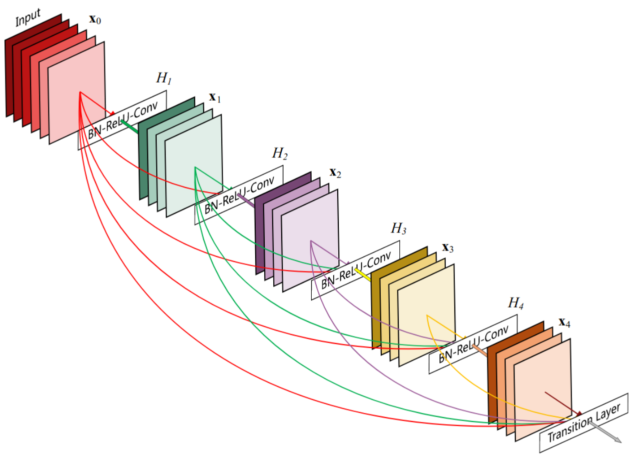


Figure 2.4: Figure from Huang et al. (2017). A 5-layer DenseNet, showing the identity mappings between every layer.

As well as the standard DenseNet, the authors introduce a modified version, referred to as DenseNet-BC. DenseNet-BC incorporates bottleneck and compression layers. The bottleneck layer introduces a simple 1x1 convolution before each 3x3 convolution, reducing the number of input feature maps. The compression layer reduces the number of feature maps in transition layers (layers between dense blocks). The transition layer has θm feature maps, where m is the number of feature maps in the previous dense block. When compressing the transition layer, $\theta < 1$ ensures the transition layer will have a reduced number of feature maps. The DenseNet-BC model tends to produce better results (4/5 best results use DenseNet-BC) while reducing the number of parameters.

DenseNet offers a significantly improved framework for training deep neural networks, while also reducing the number of parameters needed to train an effective model. By concatenating the outputs from each layer (instead of summing), any layer can directly access the feature maps of all previous layers, allowing for a much more compact model.

2.2 Neural Architecture Search

Neural architecture search is the process of finding neural network architectures through an automated search. Elsken et al. (2019) categorise NAS research into three key areas: search space, search strategy, and performance estimation strategy. The search space defines the set of possible architectures, search strategy defines the process by which an architecture is found, and performance estimation is the way in which the performance of each network is estimated. It is possible to estimate performance by training each model and evaluating it on held-out data; however, this approach is very expensive. Consequently, methods of estimating the performance of a model have been developed, avoiding the need to train and evaluate each architecture.

There are several approaches to NAS including reinforcement learning (Zoph et al., 2018) and evolutionary algorithms (Real et al., 2018). Both of these approaches can be very effective. However, they have a significant drawback - they are extremely computationally expensive. There is ongoing work to reduce the computational demand; however, both reinforcement and evolutionary approaches remain exceedingly resource intensive. As an example, Zoph et al. (2018) propose a reinforcement learning NAS approach that is 7x quicker than the previous state-of-the-art, yet still takes a considerable 2,000 GPU hours of searching. An alternative approach is to explore the search space via gradient descent. DARTS (Liu et al., 2019) relaxes the search space from discrete to continuous and then uses gradient-based optimisation to find an effective architecture. This approach is significantly less onerous than the previously mentioned methods yet still achieves very competitive results.

Formally, each DARTS cell can be considered as a directed acyclic graph. Each node, x^i, on the graph is some latent representation, such as a feature map in CNNs, and each edge, (i, j), on the graph corresponds to an operation, o (e.g. max pool). Therefore, $o^{(i,j)}$ would refer to the operation taking place along edge (i, j). Each node is computed by applying the relevant operation to each of its predecessors, and summing the result:

$$x^{(j)} = \sum_{i<j} o^{(i,j)} \left(x^{(i)}\right) \tag{2.1}$$

The process of finding a cell is split into three steps. Beginning with unknown operations on each of the edges, the model first relaxes the search space by placing multiple candidate operations along each edge. A joint optimisation procedure is then performed to optimise the mixing probabilities and network weights. Finally, the final architecture is determined by selecting the most probable operations. Cells are then stacked into the required number of layers, creating the final architecture. Cells that are at 1/3 and 2/3 of the total depth are converted into *reduction cells*, in which all operations adjacent to the input nodes have an increased stride of 2. As a result of the increased stride there are fewer parameters in the network, and a different graph for the reduction cells is typically found.

2.3 Fisher Information

Fisher information is a measure of how much information a known variable, X, contains about an unknown parameter, θ, (Lehmann and Casella, 2006) and is formally defined as in Equation 2.2.

$$I(\theta) = E_\theta \left[\frac{\partial}{\partial \theta} \log f(X;\theta) \right]^2 = \int \left(\frac{\partial}{\partial \theta} \log f(x;\theta) \right)^2 f(x;\theta) dx \tag{2.2}$$

Fisher information can be interpreted as the variance of the score, where score is the gradient of the log-likelihood with respect to the model's parameters. Setting the score to 0 gives the maximum likelihood estimation for the parameters in our model. Fisher information is the variance of this maximum likelihood estimate, which allows us to understand how certain our estimate is. Following this interpretation, Fisher information can be written as:

$$I(\theta) = \mathrm{Var}_\theta \left[\frac{\partial}{\partial \theta} \log f(X;\theta) \right] \tag{2.3}$$

where $\frac{\partial}{\partial \theta} \log f(X;\theta)$ is the derivative of the log-likelihood. Fisher information can be applied to a deep learning setting, allowing us to estimate the change in our loss function, $\mathcal{L}(\theta)$, with some change in parameters, $\mathbf{d}$. Theis et al. (2018) apply this idea to prune the channels of a network and derive an approximation for the change in loss that would occur with the removal of a channel in a neural network:

$$\begin{gathered} \mathbf{g} = \nabla \mathcal{L}(\theta), \quad \mathbf{H} = \nabla^2 \mathcal{L}(\theta) \\ \mathcal{L}(\theta + \mathbf{d}) - \mathcal{L}(\theta) \approx \mathbf{g}^\top \mathbf{d} + \frac{1}{2} \mathbf{d}^\top \mathbf{H} \mathbf{d} \end{gathered} \tag{2.4}$$

Measuring the change in loss when dropping the k^{th} parameter is then given by setting $\mathbf{d} = -\theta_k \mathbf{e}_k$, where $\mathbf{e}_k$ is a vector of zeros, except for position k, where it is 1:

$$\mathcal{L}(\theta - \theta_k \mathbf{e}_k) - \mathcal{L}(\theta) \approx -g_k \theta_k + \frac{1}{2} H_{kk} \theta_k^2 \tag{2.5}$$

Following similar work using second-order approximations, Theis et al. (2018) remove the first term, as it vanishes as we average over the dataset. In practice, including the first term also seems to reduce performance. Following further assumptions of the nature of the Hessian, H, and adjusting to get the Fisher information for entire channels, not just individual parameters, the final estimated change in loss becomes:

$$\Delta_c = \frac{1}{2N} \sum_{n=1}^{N} g_{nc}^2 \tag{2.6}$$

where g_{nc}^2 is the gradient with respect to the n^{th} data point for the channel of interest, c.

Theis et al. (2018) use this formulation for network pruning, where they compress a network by removing channels that have the lowest Fisher score. In our case, we are not interested in pruning a model but wish to expand it. Instead, we will use Fisher information to determine which cells in our network have the largest impact on the loss function and expand these cells by increasing the number of channels.

2.4 Network Distillation

Network distillation is an approach to distilling the knowledge of a deep neural network into a network with fewer parameters in such a way that the smaller model can perform similarly to the original network. To do this, there are two standard approaches: *knowledge distillation* (Ba and Caruana, 2014; Hinton et al., 2015), and *attention transfer* (Zagoruyko and Komodakis, 2016a). Knowledge distillation encourages the student to produce outputs that are similar to the teacher network, while still minimising its own loss between its output and the target labels. Attention transfer encourages the attention maps of the teacher and student at specified layers to be similar, whilst also minimising the loss between the students' output and the target labels.

2.4.1 Knowledge Distillation

Ba and Caruana (2014) laid the groundwork for knowledge distillation in their 2014 paper, which allowed the performance of deep networks to be maintained in a model with fewer parameters. Their approach is to train a student network using the outputs of the teacher network as the target labels for the student. In this work, the student network is not trained using the standard approach of cross-entropy on the outputs of a softmax layer. Instead, the student models are trained directly on the values before softmax activation; these values are referred to as *logits*, and equate to the logarithm of the predicted probabilities output by a softmax layer. By replacing the true labels with the teacher's outputs, the student can re-create the learnt representation of the teacher.

Hinton et al. (2015) built upon the work of Ba and Caruana, proposing a knowledge distillation approach that not only attempts to replicate the teacher network's outputs, but also attempts to correctly predict the original target labels. Given a teacher, $\mathbf{t} = \text{teacher}(x)$ and student $\mathbf{s} = \text{student}(s)$, the student can be trained using knowledge distillation as follows (Crowley et al., 2018):

$$\mathcal{L}_{KD} = (1-\alpha)\mathcal{L}_{CE}(\mathbf{y}, \sigma(\mathbf{s})) + \alpha T^2 \mathcal{L}_{CE}\left(\sigma\left(\frac{\mathbf{t}}{T}\right), \sigma\left(\frac{\mathbf{s}}{T}\right)\right) \tag{2.7}$$

where σ is a softmax function, α controls the ratio between the two terms, and T is a "temperature" term. A high temperature produces a softer probability distribution over classes. The first term is a standard cross-entropy loss between the students predicted output and the true labels. The second term is the cross-entropy loss between the student and teacher's predictions. To minimise this loss the student network has to correctly predict the target labels whilst also having a similar output to the teacher network.

2.4.2 Activation Based Methods

Attention transfer (Romero et al., 2015; Zagoruyko and Komodakis, 2016a) is an alternative approach to network distillation where intermediate representations at specified layers are used to guide the student network when training. Attention transfer is the approach we will use for all of our distillation experiments.

Romero et al. (2015) introduce FitNets, which propose the use of *hints*; the outputs of a teacher network are used to guide the student network at a particular layer. These hints act as a form of regularisation, limiting the student networks ability to overfit the training data. By selecting which layer to hint, the extent of the regularisation can be determined; hinting later layers applies stronger regularisation while hinting earlier layers gives the student greater flexibility. To perform this hinting, the authors proposed a loss function that encourages the output of the teacher and student networks to be similar at a specified layer, guiding the student towards the learnt representations of the teacher at that point. The loss function is formulated as follows:

$$\mathcal{L}_{HT}\left(\mathbf{W}_{Guided}, \mathbf{W}_{\mathbf{r}}\right) = \frac{1}{2}\left\|u_h\left(\mathbf{x}; \mathbf{W}_{Hint}\right) - r\left(v_g\left(\mathbf{x}; \mathbf{W}_{Guided}\right); \mathbf{w}_{\mathbf{r}}\right)\right\|^2 \tag{2.8}$$

where u_h and v_g are the teacher and student networks, respectively, up to the guided layer. $\mathbf{W}_{Guided}$ is the parameters of the guided student, $\mathbf{W}_{Hint}$ the parameters of the teacher, and $\mathbf{W}_r$ the parameters of a regressor function, r, on top of the guided layer.

As the teacher will typically be wider than the student (FitNet), a regressor is added to the layer being guided. This regressor has an output that matches the size of the hint layer.

Zagoruyko and Komodakis (2016a) build upon this idea, proposing the complete attention transfer method that is commonly used. Instead of using the outputs of each network at specific layers, attention transfer uses the attention maps of each network. The attention maps are defined by spatial maps of the activations at each layer of interest. The final loss function is then constructed as:

$$\mathcal{L}_{AT} = \mathcal{L}_{CE}(\mathbf{y}, \sigma(\mathbf{s})) + \beta \sum_{i=1}^{N_L} \left\| \frac{\mathbf{f}(A_i^t)}{\left\|\mathbf{f}(A_i^t)\right\|_2} - \frac{\mathbf{f}(A_i^s)}{\left\|\mathbf{f}(A_i^s)\right\|_2} \right\|_2 \tag{2.9}$$

with some choice of layers $i = 1, 2, ..., N_L$, and A_i representing the set of spatial maps of the activations for all of the channels in the teacher (t) or student (s).

The first term in the loss function is a standard cross-entropy loss. The second term is now encouraging the *spatial distributions* of the teacher and student activations to be similar at the specified layers. This new loss function encourages the student to pay attention to the same salient features that the teacher is attending to.

While both knowledge distillation and attention transfer are effective approaches for compressing a teacher network or enhancing the performance of a student, the original attention-transfer paper, as well as more recent work (Crowley et al., 2018; Turner et al., 2019b), suggests that attention transfer is the more effective approach. It is also possible to combine both approaches by including the second term of the knowledge-distillation loss function into the attention-transfer loss. Using both attention transfer and knowledge distillation has the effect of encouraging both the spatial distributions and outputs of the student to be similar to the teacher.

2.5 Related Work

As far as we are aware, there has been no work on developing a teacher network when given a student, making this project of particular interest. However, there has been work that does the reverse — developing a student network when given the teacher.

Crowley et al. (2018) develop an approach that they call *Moonshine*. Moonshine does not perform any reduction of the teacher through reducing depth or width, but instead replaces the convolutional operations with a cheaper alternative. The cheaper convolutions proposed are grouped convolutions and bottlenecks. The grouped-convolution approach separates the convolutions into g groups. Separating the convolutions into groups limits the convolutions by allowing channel mixing only within groups, and also reduces the number of parameters required. The authors then apply some inter-group mixing by performing a point-wise convolution after each grouped convolution. These grouped convolution are denoted as $G(g)$. The authors also define a bottleneck block, $B(b)$. The bottleneck block reduces the number of channels through a point-wise convolution, a full convolution on the reduced parameters, and finally another point-wise convolution to return the number of channels to their original size.

Turner et al. (2019b) propose an alternative approach to developing a student from the teacher, via Fisher pruning. The authors use Fisher information (Eq. 2.6) to approximate the change in error with the removal of a channel. The Fisher information can be computed across all channels, and the channels with the smallest signal are removed. A small Fisher information value indicates that a channel will have a small impact on the loss and, therefore, does not contribute much to the learning process.

Taking inspiration from both Crowley et al. (2018) and Turner et al. (2019b), BlockSwap (Turner et al., 2019a) is an approach, similar to Moonshine, that replaces convolutional blocks with a cheaper alternative, and then uses the newly created network as a student for distillation. However, unlike Moonshine, BlockSwap considers a range of different types of cheap convolutional blocks; it uses Fisher information to determine which combination of blocks is best and uses the resultant network as a student for distillation. One approach would be to test every possible combination of the different blocks and determine which gives the best performance. However, exhaustively searching through all the possible combinations can quickly become very expensive. BlockSwap takes an alternative approach; block configurations are randomly sampled and ranked by their Fisher score. The Fisher score is determined using Equation 2.6 after each block, and the score is summed across the entire network. The network configuration with the largest Fisher score is then trained and used as a student network for distillation, and the original network before applying BlockSwap is used as the teacher.

All of these approaches provide interesting and useful solutions for developing a student for distillation when a trained teacher is already available. However, they all share a common weakness, in that they all produce a student network when a teacher is already available; they do not help if all that is available is the network that is is to be deployed — i.e. the student. In the case where the only available network is the student, none of the above are of use in determining an effective teacher, particularly in cases where the student is non-standard.

2.6 Datasets

The majority of the research carried out in this project will use the popular CIFAR-10 dataset (Krizhevsky and Hinton, 2009). CIFAR-10 is a labelled subset of the 80 million tiny images dataset (Torralba et al., 2008). The dataset consists of 60000 32x32 images, with ten separate classes. The training set consists of 50000 images, and 10000 are left for the test set. CIFAR-10 is a widely used dataset that makes it very useful for comparison against previous work while also being small enough to run experiments in a period of hours, not days. Figure 2.5 shows the ten classes used in CIFAR-10, as well as example images for each class.

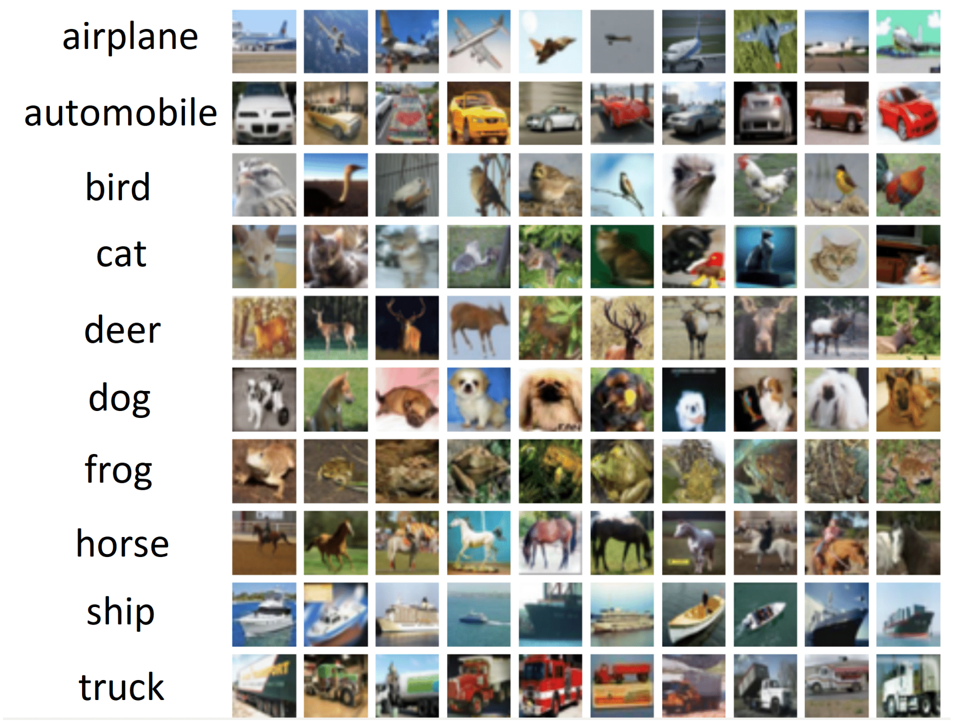


Figure 2.5: Modified figure from Krizhevsky and Hinton (2009). Example images for each of the 10 classes in CIFAR-10

Chapter 3

Methodology

As discussed in the preceding chapter, previous work has made use of Fisher information to create student networks from a teacher (Turner et al., 2019a; Turner et al., 2019b). In this work, we take a similar approach and use Fisher information to grow a teacher network from a student found through NAS.

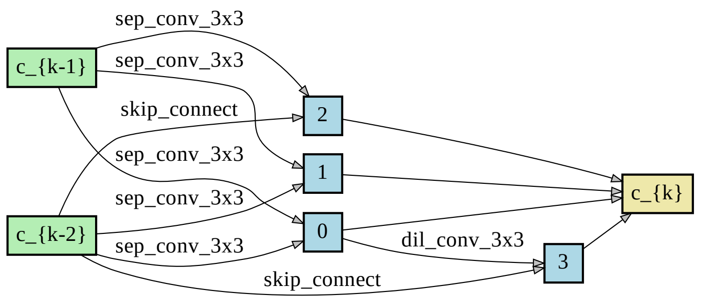
3.1 Off-The-Shelf Teacher Networks

We begin our work by investigating the performance of network distillation using a student network obtained through a DARTS search on CIFAR-10. We chose a DARTS model as it is very competitive with the state-of-the-art in NAS, is much quicker than many alternative approaches (see Section 2.2), and has an official implementation that is widely available[1]. This implementation includes many of the models found for different datasets. As a result, we do not have to do any architecture searching ourselves as models found for CIFAR-10 are readily accessible. Figure 3.1 shows the cells we used for all of our DARTS models. We will refer to models created using this cell structure as DARTS_V1 models.

The DARTS model shown in Figure 3.1 was integrated into a pre-existing network distillation codebase. In order to integrate the DARTS models, it was essential to modify them such that they returned the activations at multiple layers, as needed for attention transfer (Eq. 2.9). For each model, three activations were returned, spread evenly across the layers of the network, prior to any pooling operations.

As discussed in Section 2, we will use DenseNets and WideResNets as our off-

[1] `https://github.com/quark0/darts`



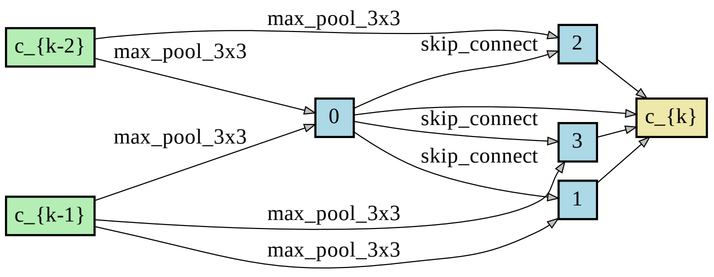


Figure 3.1: *Top:* Normal cell found for CIFAR-10. *Bottom:* Reduction cell found for CIFAR-10. Both cells were were provided by the authors so searching on CIFAR was not necessary (Figures were created using the provided **visualize.py** script).

the-shelf teacher networks. We integrated pre-trained models on CIFAR-10 [2] into the codebase, modifying them to return activations in the same manner as with the DARTS models.

The DARTS, DenseNet, and WideResNet models can now be trained via attention transfer as both a student and a teacher, giving nine possible teacher-student combinations. When using an off-the-shelf model as a teacher, a pre-trained model that is larger than the student is used. When the DARTS model is used as a teacher, that model must be trained independently before it can be used as a teacher for attention transfer or knowledge distillation. The training of the DARTS teacher follows the experimental setup described in Section 4.1.

3.2 Distillation with a Fisher Expanded Student

In this section, we introduce our two-step process for distilling with a DARTS student and expanded DARTS teacher.

[2]`https://github.com/osmr/imgclsmob`

To develop the expanded teacher, we use Fisher information as a sensitivity metric to expand our DARTS student model, creating a teacher for network distillation. Using Equation 2.6, we can approximate the change in error with the removal of each channel for each cell in our DARTS student network. Using this Fisher information it is possible to create a measure of Fisher sensitivity on a per-cell level by summing the Fisher score, Δ_c (Eq. 2.6), across all of the channels within each cell:

$$\Delta_{cell} = \sum_{1}^{C} \Delta_c \tag{3.1}$$

This idea is inspired by Turner et al. (2019a), who take the same approach to approximate the impact of each cell on the loss function. In order to compute the Fisher information for each cell, we insert a *Fisher probe* after the final operation in each cell that is updated during the backward pass, using the most recent gradients to compute the Fisher information. Turner et al. (2019a) found that the placement of the Fisher probe was inconsequential; placing the probe at the end of each cell will not be dependent on that cells specific operations and will provide consistent results. Starting from a trained student, we push a single mini-batch through our model to get the Fisher information for each cell. Turner et al. (2019a) then sum the total Fisher score across the entire network, producing a final measure of the parameters' ability to impact the loss function (this can be interpreted as the networks' ability to use its parameters effectively). In this work, we take a different approach, where we maintain our Fisher scores on a cell-level and use this to rank the ability of each cell to impact the loss. Upon ranking each cell by its Fisher score, we then decrease the group size of all applicable operations within the highest ranking cell. By default, the group size is equal to the number of channels, giving a depth-wise convolution (Sifre and Mallat, 2014). After pushing a mini-batch through the network and ranking the cells according to their Fisher score, we halve the number of groups for all operations that use grouped convolutions. Another mini-batch is then pushed through the network, and the process is repeated. If the top-ranking cell has only one group, we then reduce the number of groups in the next highest scoring cell.

Grouped convolutions are a way of reducing the number of parameters in a model, so by reducing the number of groups, we are increasing the number of parameters in cells with high Fisher scores. Reducing the number of groups in this manner is effectively enhancing the representational power of cells that are utilising their parameters most effectively. Our process of pushing a mini-batch through and growing the appropriate cell continues until the resultant model is within 2.5% of the required model size. Our process of growing the size of our network can be considered an example of *network morphism* (Wei et al., 2016) - a general approach to morphing a trained neural network while maintaining the function of the network. Algorithm 1 describes the process in more detail:

```
model = DARTS_V1-10;
while params < param goal do
    probes = model(mini-batch);
    $\Delta_{cell}$ = sum(probes);
    sort($\Delta_{cell}$);
    for i, probe in $\Delta_{cell}$ do
        if cell[i].grow() == success then
            break
        end
    end
    params = paramCount(model)
end
```

Algorithm 1: Our process for expanding cells in a model using Fisher information as a sensitivity metric. Cells with the highest Fisher score are expanded first, until they have reached 1 group (returning false). Then the next highest cell is used.

Upon obtaining a complete model through our Fisher expansion process, this new model is used as a teacher for network distillation, with the original DARTS_V1 model as the student. Specifically, this will use attention transfer to be comparable to the previous work described in Section 3.1. In order to understand the utility of our method, it should be compared to using a standard, larger DARTS model as the teacher, and also one that is expanded in a naive fashion.

Chapter 4

Experiments and Results

4.1 Experimental Setup

All experiments conducted in the following chapter use the same fundamental setup. Any changes from this setup will be mentioned explicitly where appropriate. All models are trained for a total of 200 epochs, with a learning rate of 0.1, a batch size of 64, and 16 initial channels. Models are optimised through SGD, and the MultiStepLR[1] scheduler is used to decay the learning rate at specified epoch milestones. In our case, these milestones are the 60th, 120th, and 160th epoch. At each of these epochs, the learning rate is decayed by a ratio of 0.2. The loss function used is a cross-entropy loss[2] and all models are trained and evaluated on CIFAR-10.

4.2 Off-the-shelf Teacher Networks

As discussed in Chapter 3, we begin by testing our DARTS model (Figure 3.1), a DenseNet and a WideResNet. A standard WideResNet model is used without dropout layers. We test all nine possible configurations of these models with the setup shown in Table 4.1. Each student model has approximately 7×10^5 parameters, and each teacher, approximately 1.4×10^6 parameters, making the student models about half the size of each teacher model. Combinations of each of the models in Table 4.1, were then trained using attention transfer. These experiments included each model with itself (e.g. dense teacher + dense student), giving a total of 9 combinations. Each experiment is performed three times to ensure that our results are consistent. The error

[1]`https://pytorch.org/docs/stable/optim.html`
[2]`https://pytorch.org/docs/stable/nn.html`

bars allow for an understanding of how much variation we see across results and are given by the mean error ± standard deviation.

Model	Teacher/Student	Parameters	Model Details
DARTS	Teach	1354980	DARTS_V1-25
	Student	774260	DARTS_V1-10
DenseNet	Teach	1542682	DenseNet-BC-40 (k = 36)
	Student	769162	DenseNet-BC-100 (k = 12)
WRN	Teach	1467610	WRN-28 (k = 2)
	Student	691674	WRN-16 (k = 2)

Table 4.1: Model details for performing distillation using combinations of DARTS, WideResNet and DenseNet. The parameter, k, refers to the models widening factor (WRN) or growth rate (DenseNet). The additional number in the model names refers to the number of layers in the model (DARTS_V1-10 has 10 layers). Each model was selected such that the number of parameters in the teachers was approximately twice that of the students.

Teacher	Student	Top-1 Error	Top-5 Error
DenseNet	DenseNet	6.3000 ± 0.35	0.1833 ± 0.02
DenseNet	WRN	6.2467 ± 0.12	0.2000 ± 0.04
DenseNet	DARTS	8.4600 ± 1.30	0.24 ± 0.04
WRN	DenseNet	5.7467 ± 0.33	0.1767 ± 0.02
WRN	WRN	6.1567 ± 0.17	0.2200 ± 0.02
WRN	DARTS	8.5433 ± 0.14	0.2233 ± 0.02
DARTS	DenseNet	6.7833 ± 0.02	0.2233 ± 0.01
DARTS	WRN	6.6767 ± 0.42	0.2067 ± 0.01
DARTS	DARTS	7.7900 ± 0.30	0.2100 ± 0.01

Table 4.2: Experimental results of teacher-student pairings on CIFAR-10 using attention transfer. DARTS students typical under-perform, and teacher-student combinations appear to be most effective when both models have similar architectures. Highlighted in blue is the best performing pairing for each student. Each experiment was ran three times.

Table 4.2 shows the results for our different teacher-student pairings. Generally, DARTS students appear to perform poorly, particularly when paired with a WRN or

DenseNet. Both WRN and DenseNet make good pairings for each other, with the best performing pairing being a WRN teacher with a DenseNet student. DARTS students performs best when a DARTS teacher is also used.

4.3 Distilling with a Fisher Expanded Student

After experimenting with combinations of our DARTS_V1 model and pre-trained, off-the-shelf networks, we then perform our Fisher expansion and distillation using the Fisher-expanded DARTS_V1-10 model as the teacher, and original DARTS_V1-10 model as the student.

Model	Parameters	Top-1 Error	Top-5 error
DARTS_V1-25	1354980	6.85	0.16
DARTS_V2-10-f-untrained	1459347	N/A	N/A
DARTS_V2-10-u	1428452	7.32	0.23
DARTS_V2-10-f	1447460	6.91	0.2

Table 4.3: Details of each of the teacher models used for our Fisher expansion experiments. Error bars are not shown as these are the individual models used for distillation. The number of parameters for all models is within 2.5% of 1.4×10^6. **f** refers to a Fisher expanded model. **u** refers to a uniformly expanded model. **untrained** denotes a model that has not been trained before being used as a teacher.

We begin with a standard, DARTS_V1-10 model, and repeatedly push individual CIFAR-10 mini-batches through the model and decrease the group size of the highest-scoring cells, as described in Chapter 3. The expansion process continues until our new model is within 2.5% of 1.4×10^6 parameters, ensuring the approximate 2:1 teacher-student size ratio is maintained. We refer to all base-models generated by reducing the number of groups as DARTS_V2. Generally, DARTS_V2 models must then be trained to be used to teach a student model with attention transfer. The details of all of the teacher models used in the subsequent Fisher expansion experimentation, as well as their performance on CIFAR-10, is shown in Table 4.3.

In order to gauge the effectiveness of our Fisher expansion approach, we also grow the DARTS_V1-10 model uniformly (DARTS_V2-10-u), again reducing the number of groups until the desired number of parameters is reached. The resultant model is also trained using the same experimental setup, and used for distilling with a DARTS_V1-

10 student. Furthermore, we have experimented with using a teacher network that is scaled up using our Fisher expansion process but has not been trained at all. By experimenting with an untrained, Fisher expanded teacher (DARTS_V2-10-f-untrained), we can see how informative Fisher information is when trying to model a teacher network. Comparisons between DARTS_V2-10-f-untrained, DARTS_V2-10-f, DARTS_V2-10-u and DARTS_V1-25 should give a comprehensive understanding of the utility of our approach. The results of using each of these teacher models for distillation with a DARTS_V1-10 student are shown in Table 4.4.

Teacher	Student	Top-1 Error	Top-5 Error
N/A	DARTS_V1-10	8.4700 ± 1.11	0.2200 ± 0.04
DARTS_V1-25	DARTS_V1-10	7.7900 ± 0.30	0.2100 ± 0.01
DARTS_V2-10-f-untrained	DARTS_V1-10	7.1050 ± 0.09	0.1400 ± 0.03
DARTS_V2-10-u	DARTS_V1-10	6.4233 ± 0.17	0.1833 ± 0.06
DARTS_V2-10-f	DARTS_V1-10	6.4033 ± 0.08	0.1333 ± 0.03

Table 4.4: Results of our expanded models. Using a teacher developed through our Fisher expanded approach gives the best results with little variation in performance. However a uniformly expanded model is also competitive. Interestingly, the DARTS_V2-10-f-untrained model is able to improve the performance of the student and has a very competitive top-5 error. Each experiment was ran three times.

Most notably, the best performing performance is achieved by combining a DARTS_V2-10-f teacher with the DARTS_V1-10 student, obtaining both the best top-1 and top-5 error. This result indicates that our Fisher expansion approach is very effective in developing a suitable teacher from the student. The results also show that whilst not as effective as a DARTS_V2-10-f, a simple uniform expansion of the student gives very competitive results (DARTS_V2-10-u). Furthermore, the DARTS_V2-10-f-untrained model gives a reasonably poor performance, however, still outperforms both a DARTS_V1-25 teacher, and a DARTS_V1-10 student trained without distillation.

Whilst using a DARTS_V2-10-u teacher does give competitive results, there is much more variation across runs (shown by the error bars) than when using a DARTS_V2-10-f teacher. This variation is seen in both the top-1 and top-5 error, suggesting that the DARTS_V2-10-f teacher may produce more consistent results. The DARTS_V2-10-u also has a poor top-5 error, despite top-1 error being strong.

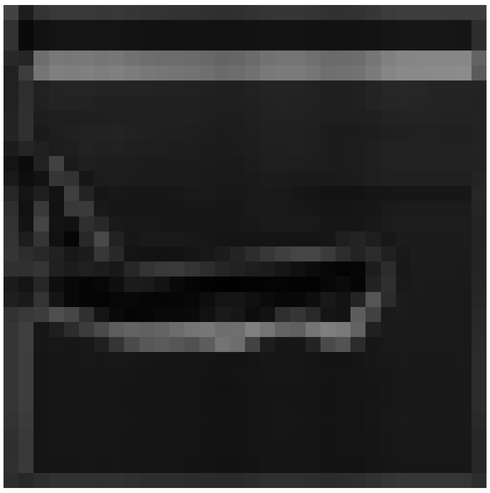
Chapter 5

Discussion

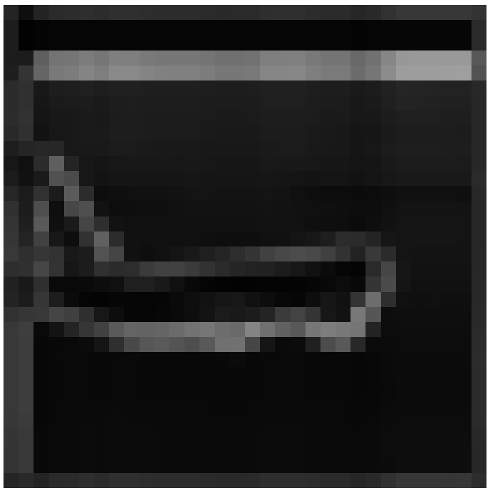
5.1 Distilling With Off-The-Shelf Networks

5.1.1 Teacher-student Pairings

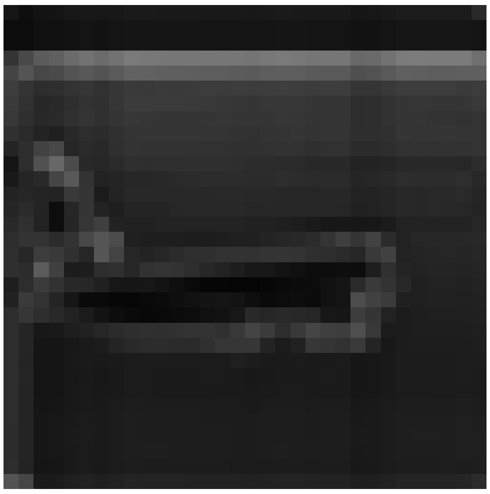
The results shown in Table 4.2 provide much insight into the application of DARTS models — as well as the other models used — in a distillation setting. In general, it seems that the best teacher for any student is simply a scaled-up version of the student. This pattern of good distillation performance when using the same student and teacher architectures was also observed by Crowley et al. (2018). One possible explanation is that when both the teacher and student are using the same underlying architecture, they are more likely to make similar predictions, and may well have similar activations within each model. When using attention transfer (Eq. 2.9), the distillation process tries to encourage the students' attention map (spatial map of the activations) to be similar to the teachers' attention map. If the activations are too dissimilar, it may be difficult to guide the student towards an effective representation. When the student and teacher models are similar, the teacher may be able to guide the student in a more nuanced and effective manner. In cases where the teacher activations are significantly different from the student activations, on a local level, the teacher's signal may be somewhat contradictory to the signal obtained from the student. By visualising the attention maps of our networks, it is possible to examine the effect of different teachers on a student (Figure 5.1).



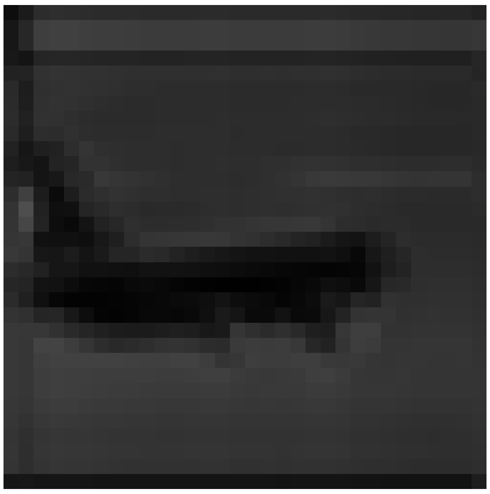
No teacher, DenseNet student



DenseNet teacher, DenseNet student



WRN teacher, DenseNet student



DARTS teacher, DenseNet student

Figure 5.1: Visualisations of the activations for each DenseNet student model. The DenseNet teacher reinforces the representation learnt by the student. As the teacher becomes increasingly dissimilar from the student the learnt representation becomes much less clear, resulting in a vague blur when a DARTS teacher is used. The activations are taken from the middle of the network. Specifically, an activation is taken at each point the cell output reduces in size. These images are the 2nd of three points where that occurs.

Figure 5.1 shows visualisations of the activations for multiple DenseNet configurations when a CIFAR-10 image of a plane is passed through each model. The visualisations offer a lot of insight into the impact of different teacher architectures on the student; it is clear that when the teacher and student architectures are the same, the network is learning something very similar to that which the student would learn independently. By incorporating a teacher of the same architecture, that learnt representation is just reinforced, strengthening the ability of the teacher to recognise images. As the teacher architecture diverges from the student architecture, the representation is instead blurred, giving increasingly less distinct representations as the teacher goes from DenseNet to WRN to DARTS. As mentioned previously, this blurring may be a result of contradictory signals from the teacher and student, making it challenging to learn an effective representation.

Interestingly, Mirzadeh et al. (2019) have a somewhat related hypothesis in their work on knowledge distillation (see Section 2). The authors find that given a student network of a fixed size, an arbitrarily large teacher cannot be used and that the gap between student and teacher must not be too large. Their work is on knowledge distillation and considers the differences in networks with regards to the number of parameters, rather than differences in architecture. Nonetheless, these findings do support our hypothesis that teacher and student networks cannot vary too much from each other and still perform well.

However, there is one exception to this trend within our experiments. In the case of a WRN teacher and a DenseNet student, the average top-1 error is 5.75%. This result is noticeably lower than when a WRN student is used with a WRN teacher (6.16%), although the standard deviation in the WRN-DenseNet case is notably more significant. This excellent performance may be a result of DenseNet's connections to and from every layer in the network, and concatenation of layer outputs, which reduce the number of parameters required for the model to perform effectively. Consequently, the student DenseNet may simply be a much more capable model than the WRN student, allowing it to obtain a better result. However, it is still not clear why the WRN teacher gives notably better results for a DenseNet student than a DenseNet teacher. The DenseNet authors do note in later experiments[1] that an effective dense model can take a very long time to train using their standard setup, and suggest using a Wide-DenseNet instead, with fewer layers and a high growth rate to achieve effective results with less training. As all of our experiments were only limited to 200 epochs, it may be worth experimenting with wider, shallower models to see if the WRN teacher and DenseNet student combination maintains its performance. Additionally, further experimentation would be needed to reduce the variation of results that are seen in this specific case. However, as DenseNet and WRN both have a very similar underlying model based upon the original ResNet architecture, these results still support our overall hypothesis

[1] `https://github.com/liuzhuang13/DenseNet`

that similar models work best for distillation, explaining why the DARTS models give significantly worse results when combined with a DenseNet or WRN teacher.

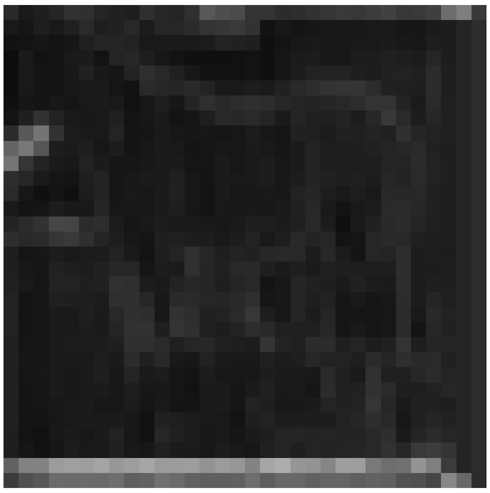
5.1.2 Distilling with Non-Standard Networks

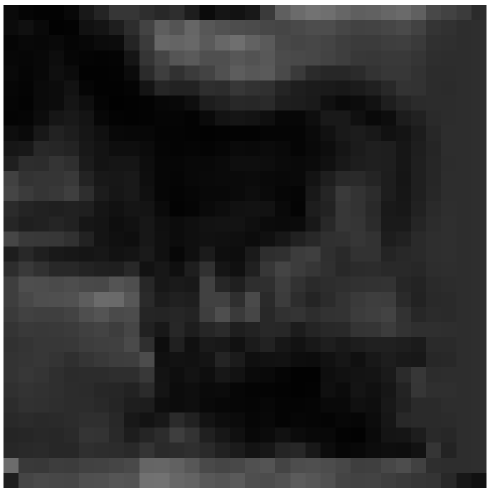
Most notably, the results shown in Table 4.2 provide some insight into the effectiveness of a non-standard model in a distillation setting. There are two clear takeaways with respect to the DARTS models. Firstly, DARTS students (DARTS_V1-10) tend to perform significantly worse than WRN and DenseNet students, irrespective of which teacher they were trained with. Table 4.2 shows that the best performing DARTS student model has a top-1 error of 7.79, whereas WRN and DenseNet students are able to reach top-1 errors of 6.1567 and 5.7467, respectively. Secondly, when used as a teacher, the DARTS models give reasonably competitive results. Table 4.2 shows that a DARTS teacher (DARTS_V1-25) is able to reach a top-1 error below 7 when paired with either a DenseNet or WRN student.

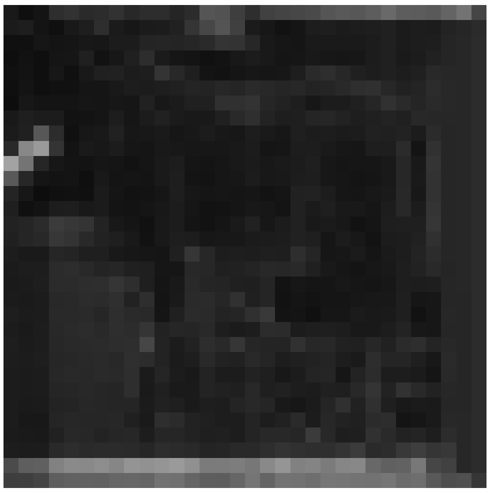
The generally poor performance of DARTS models when used as a student may be a result of the models' particularly non-standard architecture. As a result of the substantial differences in architecture when a DARTS student is used, it be may difficult for the teacher network to accurately guide the student using attention transfer. These large differences will be most prevalent when using a DARTS student, which is very different from the ResNet base shared by both WRN and DenseNet. This hypothesis is supported by the fact that training a DARTS student using a DARTS teacher gives significantly better results when compared to using a WRN or DenseNet teacher with the DARTS student. Even so, the DARTS teacher-student setup still has the worst performance of all other experiments with a DARTS teacher. It seems that similarity in architecture between the teacher and student is most important when the student is non-standard. However, there may also be something fundamental about the DARTS model that causes this reduction in performance that is not the result of substantial differences between teacher and student architectures. This inherently poor performance may simply be a result of the small size of the DARTS models; the original DARTS paper trained their CIFAR-10 model for 600 epochs, with 64 channels and 20 cells. Both the teacher and student models used in our experiments were smaller than that in the original paper, suggesting that in order to utilise the DARTS architecture most effectively, the model must be larger and trained for a more extended period. However, further experimentation is required to either confirm or deny this hypothesis.

5.2 Distilling with a Fisher Expanded Student

Table 4.4 shows results for our experiments using a Fisher expanded DARTS_V1-10 model as the teacher using attention transfer. Our Fisher expansion approach gave the best results for both top-1 and top-5 errors, suggesting that it is an effective approach for expanding a student model. Interestingly, a uniformly expanded teacher also provides competitive results, although with a much higher standard deviation as well as a higher top-5 error than the Fisher expansion approach.


No teacher, DARTS_V1-10 student


DARTS_V1-25 teacher, DARTS_V1-10 student


DARTS_V2-10-u, DARTS_V1-10 student

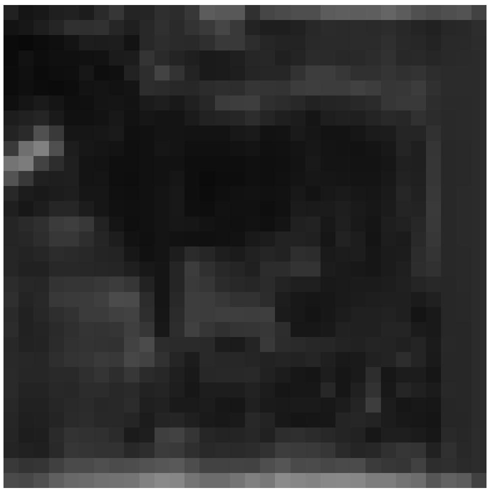

DARTS_V2-10-f teacher, DARTS_V1-10 student

Figure 5.2: Visualisations of the activations for each DARTS_V1-10 student setup. The uniform and fisher expansions appear to be learning something similar to the DARTS_V1-10 student, with the fisher expanded teacher giving the most distinct representation. The DARTS_V1-25 teacher appears to be learning something different to the rest of the models, explaining its poor performance.

5.2.1 Visualising DARTS Teacher-Student Combinations

Figure 5.2 shows attention maps for each of the experiments shown in Table 4.4 on a randomly selected CIFAR-10 image of a horse. As all of these models have a teacher-student pairing that uses the same general architecture, there is noticeably less variation in the models' learning than in the attention maps shown in Figure 5.1. It seems that the student trained using a Fisher expanded DARTS teacher (DARTS_V2-10-f) has learnt the most distinct representation, emphasising the outline of the horse while ignoring some of the details in the legs. The student trained using a uniform expanded teacher (DARTS_V2-10-u) has also learnt a similar representation, although the horse outline is not as distinct as in the Fisher expanded case, and some of the detail in the horse is also lost. When using a DARTS_V1 teacher and student, the horse-like representation is still visible; however, it is significantly distorted. These visualisations support the results in Table 4.4, showing that similar teacher-student pairings give the best performance, and in particular, Fisher expansion is the most effective approach. Comparison between the visualisations and results in Table 4.4 suggests that a learnt representation that focuses on the outlines of key shapes in an image is probably most effective, and by picking up on small details, such as in the no teacher, DARTS_V1-10 student case, the model's ability to generalise may be impacted. As the DARTS_V1-25 teacher has significantly more layers than the DARTS_V1-10 student, it is also the most different from the original architecture, explaining the poor results and distorted image.

5.2.2 Evaluating our Fisher Expansion Approach

It is interesting that while the Fisher expansion approach outperformed all the other models, using a uniform expansion also gave very competitive results. Examining the Fisher expansion process on our DARTS_V1-10 student shows that the Fisher expansion process grew many of the earlier cells in the network, suggesting that these cells have the most impact on the loss function, and, therefore, the models' ability to learn an effective representation. The uniform expansion process starts from the first cell in the network, reduces the number of groups, and moves to the next, reducing the number of groups until the desired number of parameters is reached. As a result of the uniform expansion starting from the first cell, and the Fisher information indicating that cells early in the network are important, there is considerable overlap in the choice of cells between the two approaches. It seems likely that Fisher expansion would give a much larger performance gap when much larger models are being used, as the Fisher expansion can be much more nuanced concerning the cells that are being expanded. In order to test this, further experimentation is necessary using larger models. Additionally, the effectiveness of our Fisher expansion approach can be further evaluated by comparing to an alternative, naive expansion method, such as a random expansion. Using a random expansion would minimise the likelihood of there being a lot of overlap between the Fisher and naive expansion methods. Additionally, random search methods have been shown to be quite powerful for NAS, consistently beating the start of the art (Li and Talwalkar, 2019; Sciuto et al., 2019), and may, therefore, have some utility in our case.

5.2.3 Distillation with an Untrained Fisher-expanded Model

Another interesting finding of our experimentation is that when using an untrained, Fisher expanded teacher, the resultant model is capable of outperforming a DARTS_V1-10 model trained independently without any distillation, as well as a model distilled using a DARTS_V1-25 teacher. While using an untrained teacher does not give competitive results, this finding follows from our previous discussion by highlighting that it is the architecture that is most important for effective distillation. This result takes our findings a step further, suggesting that the performance of the teacher network itself is not that important. As long as the architecture is compatible with the student, it is possible to improve the performance of a smaller model using attention transfer. Gaier and Ha (2019) have similar findings in a non-distillation setting, showing that it is possible to have an effective model, regardless of the values of the models' parameters and conclude that the architecture alone is often enough to achieve reasonable performance. Their work uses a random initialisation and can find models that are effective on multiple basic reinforcement learning tasks without any further training of the weights. In order to evaluate this result properly, more work is required, comparing the use of untrained versions of many of the models used in this work.

Chapter 6

Conclusions

In this project, we have explored the impact of different teacher-student pairings for network distillation, as well as proposed an effective approach for developing a teacher network from a given student.

The results of our exploration of teacher-student pairings suggest that off-the-shelf networks only make effective teachers when they are suitably similar to the student network being trained. When the teacher and student are both from the same architecture, the resultant student tends to perform well. When a non-standard student generated through neural architecture search is combined with an off-the-shelf teacher architecture, performance degrades significantly. However, when paired with a deeper version of that same non-standard model, performance is notably improved, although it is still not competitive. This finding shows that architecture similarity is very important when determining an effective teacher-student pairing, and completes the first objective of our project by showing that off-the-shelf teacher architectures are only effective when the student is of a similar architecture. The results also suggest that there may be better approaches to developing a teacher architecture for non-standard students than merely scaling the depth of the network.

The second objective of this project was to identify an approach to modelling a teacher network when given only the student architecture. The results from our teacher-student pairing experimentation suggest that a depth-wise scaling of the student is better than using some off-the-shelf alternative; however, it still does not provide competitive results. Consequently, we proposed a novel approach to growing the number of channels in a student network that we call Fisher expansion. The Fisher expansion algorithm uses the Fisher information of cells in a network to identify which cells have the most significant impact on the loss function and reduces the number of groups in those cells, effectively increasing the number of channels within the cell. Using a teacher derived through a Fisher expansion of a non-standard student gave a significant boost in performance over a depth-wise scaling of the student. We compared our Fisher expansion approach to a simple uniform scaling of the channels in the student and demonstrated that Fisher expansion offers the best performance. However, the uniform expansion was also notably more effective than a depth-wise scaling of the student, suggesting that while Fisher expansion is most effective, a general approach of increasing the number of channels in the student network to create a teacher is preferable to any depth-wise scaling. This finding follows on from our work on the first objective, demonstrating that a teacher network that is most similar to the student is likely to result in the best model. A teacher that is notably deeper than the student may have very different activations, making it difficult for the attention transfer process to guide the student effectively. By keeping the depth constant and, instead, scaling the number of channels, the activations are more likely to remain similar, giving an increase in performance. We verify this intuition by examining the attention maps of a range of models and demonstrate that using a teacher architecture similar to student reinforces the learnt representation of the student and using a different teacher architecture to the student appears to distort the learnt representation.

Furthermore, we have preliminary results that show that using an untrained, Fisher expanded teacher is more effective than training the student independently, or training the student using a teacher obtained through a depth-wise expansion of the student. This finding supports all of our previous discussion by demonstrating that what is most important when determining a teacher is the models' architecture. Even when the teacher is entirely untrained, an effective architecture can aid the learning process of the student.

To conclude, we demonstrate that teacher-student pairings are only effective when both the teacher and student have very similar architectures. We also show that our proposed Fisher expansion algorithm is an effective approach to modelling a teacher network when given a non-standard student.

6.1 Further Work

Following on from this project, there are many directions that may allow for a further understanding of network distillation, particularly when using non-standard models developed through neural architecture search.

Firstly, as a result of the very competitive performance of a uniform expansion of the student, it would be interesting to understand how effective a uniform expansion can be, and in which cases our Fisher expansion approach is notably better. As discussed in Section 5.2.2, both the Fisher and uniform expansions similarly grew the model in our specific case. As our student models were quite small, the majority of early cells were expanded by each approach. By experimenting with much larger models, the Fisher expansion will likely be able to able to make more nuanced cell expansions, whereas the uniform expansion will blindly expand all cells equally. Consequently, the Fisher expanded model will likely perform better in this case.

Additionally, it may be interesting to experiment with Fisher expansion on the off-the-shelf networks in order to test how well Fisher expansion generalises to network distillation as a whole, and not just the non-standard setting. Our results suggest that a channel-wise scaling of the network is much better than depth-wise scaling. With this in mind, it may be possible to outperform our off-the-shelf results through the use of a teacher derived through a Fisher expansion. As well as experimenting with off-the-shelf networks, experimentation with alternative NAS models, such as EfficientNet (Tan and Le, 2019), may be informative in measuring the generalisability of our approach.

It may also be possible to explore more nuanced equivalents of our Fisher expansion by using the Fisher information on the level of individual channels in the network. Similar to Fisher Pruning, this would find the channels that are most impactful on the loss function and create multiple copies of that same channel. Growing individual channels would allow the weights of the network to be maintained, and the pruning methodology of prune, fine-tune, prune, could be applied.

Furthermore, we experimented with using an untrained, Fisher expanded model as the teacher and found that it offers an improvement over a student trained without distillation. More work should be carried out investigating the use of untrained teacher models to determine how widely these findings can be generalised, as well as trying to understand further why an untrained model can aid the student in some way.

Finally, the best performing teacher-student pairing was that of a WideResNet teacher and DenseNet student. Further work should be carried out with this pairing to understand why, in this case, a subtly different teacher to the student was able to outperform using the same architecture for both teacher and student. Further work should also be carried out to test the generalisability of our approach by testing Fisher expansion on larger datasets, such as ImageNet Deng et al. (2009), on alternative tasks, such as semantic segmentation, as well as with knowledge distillation.

Bibliography

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*, pages 2654–2662.

Bianchini, M. and Scarselli, F. (2014). On the complexity of shallow and deep neural network classifiers. In *ESANN*.

Collobert, R., Puhrsch, C., and Synnaeve, G. (2016). Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv:1609.03193*.

Crowley, E. J., Gray, G., and Storkey, A. J. (2018). Moonshine: Distilling with cheap convolutions. In *Advances in Neural Information Processing Systems*, pages 2893–2903.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Denil, M., Shakibi, B., Dinh, L., De Freitas, N., et al. (2013). Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born again neural networks. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1607–1616, Stockholmsmssan, Stockholm Sweden. PMLR.

Gaier, A. and Ha, D. (2019). Weight agnostic neural networks. `https://weightagnostic.github.io`.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708.

Ioannou, Y. (2017). A tutorial on filter groups (grouped convolution). `https://blog.yani.io/filter-group-tutorial/`.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lehmann, E. and Casella, G. (2006). *Theory of Point Estimation (Springer Texts in Statistics)*. Springer.

Li, L. and Talwalkar, A. (2019). Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*.

Li, Y., Chen, Y., Wang, N., and Zhang, Z. (2019). Scale-aware trident networks for object detection. *arXiv preprint arXiv:1901.01892*.

Liu, H., Simonyan, K., and Yang, Y. (2019). DARTS: Differentiable architecture search. In *International Conference on Learning Representations*.

Mirzadeh, S.-I., Farajtabar, M., Li, A., and Ghasemzadeh, H. (2019). Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *arXiv preprint arXiv:1902.03393*.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2018). Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In *Proceedings of International Conference on Learning Representations*.

Sciuto, C., Yu, K., Jaggi, M., Musat, C., and Salzmann, M. (2019). Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

Sifre, L. and Mallat, S. (2014). Rigid-motion scattering for image classification. *Ph. D. dissertation*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114, Long Beach, California, USA. PMLR.

Theis, L., Korshunova, I., Tejani, A., and Huszár, F. (2018). Faster gaze prediction with dense networks and fisher pruning. *arXiv preprint arXiv:1801.05787*.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970.

Turner, J., Crowley, E. J., Gray, G., Storkey, A., and O'Boyle, M. (2019a). BlockSwap: Fisher-guided block substitution for network compression. *arXiv preprint arXiv:1906.04113*.

Turner, J., Crowley, E. J., Radu, V., Cano, J., Storkey, A., and O'Boyle, M. (2019b). Distilling with performance enhanced students. *arXiv preprint arXiv:1810.10460*.

Wei, T., Wang, C., Rui, Y., and Chen, C. W. (2016). Network morphism. In *International Conference on Machine Learning*, pages 564–572.

Zagoruyko, S. and Komodakis, N. (2016a). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*.

Zagoruyko, S. and Komodakis, N. (2016b). Wide residual networks. In Richard C. Wilson, E. R. H. and Smith, W. A. P., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press.

Zhu, Y., Sapra, K., Reda, F. A., Shih, K. J., Newsam, S., Tao, A., and Catanzaro, B. (2019). Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8856–8865.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710.