IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 13, 2022, accepted June 5, 2022, date of publication June 13, 2022, date of current version June 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3182659

Methods for Pruning Deep Neural Networks

SUNIL VADERA AND SALEM AMEEN
School of Science, Engineering and Environment, University of Salford, Salford M5 4WT, U.K.
Corresponding author: Sunil Vadera (s.vadera@salford.ac.uk)

ABSTRACT This paper presents a survey of methods for pruning deep neural networks. It begins by categorising over 150 studies based on the underlying approach used and then focuses on three categories: methods that use magnitude based pruning, methods that utilise clustering to identify redundancy, and methods that use sensitivity analysis to assess the effect of pruning. Some of the key influencing studies within these categories are presented to highlight the underlying approaches and results achieved. Most studies present results which are distributed in the literature as new architectures, algorithms and data sets have developed with time, making comparison across different studied difficult. The paper therefore provides a resource for the community that can be used to quickly compare the results from many different methods on a variety of data sets, and a range of architectures, including AlexNet, ResNet, DenseNet and VGG. The resource is illustrated by comparing the results published for pruning AlexNet and ResNet50 on ImageNet and ResNet56 and VGG16 on the CIFAR10 data to reveal which pruning methods work well in terms of retaining accuracy whilst achieving good compression rates. The paper concludes by identifying some research gaps and promising directions for future research.

INDEX TERMS Deep learning, neural networks, pruning deep networks.

I. INTRODUCTION

Deep learning and its use in high profile applications such as autonomous vehicles [1], predicting breast cancer [2], speech recognition [3] and natural language processing [4] have propelled interest in Artificial Intelligence to new heights, with most countries making it central to their industrial and commercial strategies for innovation.

Although there are different types of architectures [5], deep networks typically consist of layers of neurons that are connected to neurons in preceding layers via weighted links. Another characteristic, which is considered central to their predictive power [6], is that they have a large number of parameters that need to be learned, with networks such as ResNet50 [7] having more than 25 million parameters and VGG16 [8] having more than 138 million weights. An obvious question, therefore, is to ask whether it is possible to develop smaller, more efficient networks without compromising accuracy? One direction of work aimed at addressing this question has been to first train a large network and then to prune and fine-tune a network. Although methods for pruning shallow neural networks were proposed in the 1980s and 90s [9]–[11], recent advances in deep learning and its potential for applications in embedded systems has led to an increasing number and variety of algorithms for pruning deep neural networks. Hence, this paper presents a survey of recent work on pruning neural networks that can be used to understand the types of algorithms developed, appreciate the key ideas underpinning the algorithms and gain familiarity with the major approaches and issues in the field. The paper aims to achieve this goal by presenting the progressive path from the earlier algorithms to the recent work, categorising algorithms based on the approach used, contrasting the similarities and differences between the algorithms and concluding with some directions for future research.

The studies on pruning methods all carry out empirical evaluations that compare the performance of algorithms on different architectures and benchmark data sets. These evaluations have evolved as new deep learning architectures have developed, as new data sets have become available and as new pruning algorithms have been proposed. This paper also provides a useful resource that brings together the reported results in one place, allowing researchers to quickly compare the reported results on different architectures and data sets.

The survey identified over 150 studies on pruning neural networks, which can be categorised into the following eight groups based on the underlying approach used:

1) **Magnitude based pruning methods** [12]–[15], which are based on the view that the saliency of weights and

The associate editor coordinating the review of this manuscript and approving it for publication was Shenghong Li.

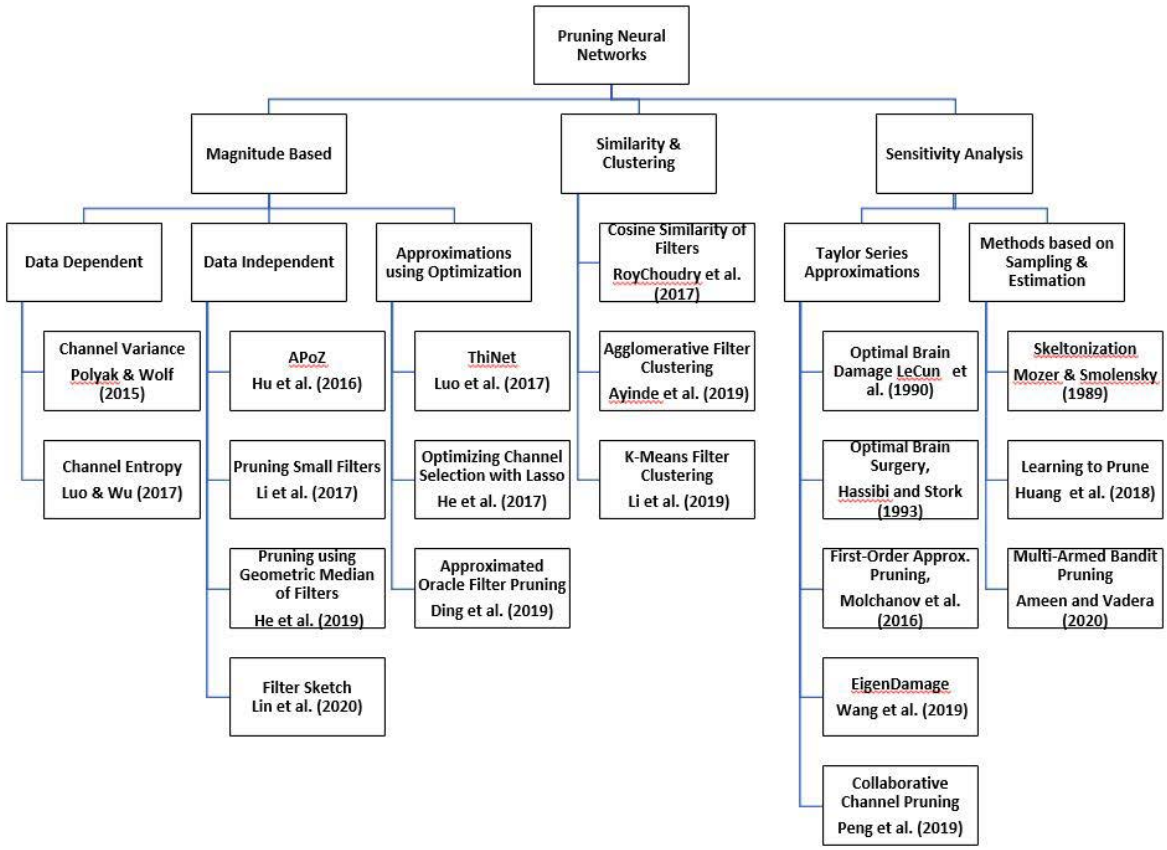

FIGURE 1. A selection of pruning methods grouped in terms of the approach adopted.

neurons can be determined by local measures such as their magnitude.

2) **Similarity and clustering methods** [16]–[21], which aim to identify duplicate or similar weights which are redundant and can be pruned without impacting accuracy.
3) **Sensitivity analysis methods** [9], [22]–[27], that assess the effect of removing or perturbing weights on the loss and then remove a proportion of the weights that have least impact on accuracy.
4) **Knowledge distillation methods** [28]–[31], which utilise the original model, termed the Teacher, to learn a more compact new model called the Student.
5) **Low rank methods** [27], [32], [33], that factor a weight matrix into a product of two smaller matrices which can then be used to perform an equivalent function more efficiently than the single larger weight matrix.
6) **Quantization methods** [34]–[39], which are based on using quantization, hashing, low precision and binary representations of the weights as a way of reducing the computations.
7) **Architectural design methods** [40]–[46], that utilise intelligent search and reinforcement learning methods to generate neural network architectures.
8) **Hybrid methods** [47]–[49], which utilise a combination of methods aimed at taking advantage of the cumulative compressing effects of the different types of methods.

Table 1 classifies over 150 studies identified by the survey into the 8 categories, enabling researchers working on a particular type of method to locate related studies. Given the range of studies, and availability of surveys already covering some of the above categories, this paper focuses on recent algorithms in the first three categories for pruning. Reed [11] provides an excellent survey of pruning methods prior to the deep learning era. Readers interested in the use of quantization, low rank and knowledge distillation methods are referred to the survey by Lebedev *et al.* [50] and readers interested in architectural design methods are referred to the comprehensive survey by Elsken *et al.* [51]. Pruning networks is just one step in developing efficient models and a recent survey by Menghan [52] summarises the full range of methods, from use of quantization and learning, to the available software and hardware infrastructure for efficient deployment of models. Another important direction of work, worthy of a survey in its own right, and not in the scope of this paper, is the use of variational Bayesian methods for regularization [53]–[59].

Fig. 1 shows a selection of the methods covered in greater detail in this survey and includes a sub-categorization of

TABLE 1. Categorisation of studies on pruning.

Magnitude based pruning methods	
1988-91	Kruschke [10], Hanson and Pratt [122], Weigend [13], Weigend et al. [14]
2011	Graves [123]
2015-2016	Han et al. [64], Polyak and Wolf [79], Guo et al. [65], Hu et al. [84], Wen et al. [58]
2017	Aghasi et al. [124], He et al. [94],Li et al. [76], Liu et al. [92], Luo and Wu [83], Luo et al. [88], Wang et al. [95] Zhu and Gupta [125]
2018	Chen et al. [126], He et al. [89], Huang and Wang [55], Lee et al. [127] Li et al. [128], Liu and Liu [129], Luo and Wu [96], Qin et al. [130] Yazdani et al. [131], Ye et al. ([97], Yu et al. [90], Zhang et al [132], Zhou et al. [15]
2019	Ding et al. [91], [133], Dettmers and Zettlemoyer [134], Frankle et al [66], [68] Gui et al. [135], He et al. [87], Helwegen et al. [136], Hou et al. [137], Lee et al. [138], Li et al. [139] Liu et al. [67], [140], Mittal et al. [119], Morcos et al. [69], Song et al. [141], Xu et al. [142], Yu et al. [71], Zhao et al. [59], Zhou et al. [143], Zhu et al. [144]
2020	Hubens et al. [70], Kim et al. [145], Li et al. [146], Lin et al. [56], [85], Niu et al. [147]
Similarity and clustering methods	
2017-2018	RoyChowdhury et al. [19], Dubey et al. [148], Son et al. [99]
2019	Ayinde et al. [78], Ding et al. [149], Li et al. [18], [100]
2021	Lin et al. [150]
Sensitivity analysis methods	
1988	Mozer and Smolensky [9]
1990-1993	LeCun et al. [22], Hassibi et al. [23], [24], [151]
2006-2009	Xu and Ho [152], Endisch et al. [153], [154]
2016	Cohen et al. [25], Grosse and Martens [107], Molchanov et al. [105]
2017	Ameen [155], Anwar et al. [156], Dong et al. [111], Guo and Potkonjak [157], Neklyudov et al. [114]
2018	Lin et al. [27], Bao et al. [115], Carreira-Perpinan and Idelbayev [158], Jiang et al. [110], Huang et al. [93], Huynh et al. [159], Zhuang et al. [112]
2019	Chen et al. [160],Deng et al. [161],Jin et al. [162], Lee et al. [26] Li et al. [163], Molchanov et al. [106], Peng et al. [108], Qin et al. [164], Wang et al. [101], Xiao et al. [165]
2020	Ameen and Vadera [118]
Knowledge distillation methods	
2006	Bucilua et al. [28]
2014-2015	Romero et al. [82], Hinton et al. [29],
2017	Gregor Urban et al. [30]
2019	Bao et al. [166], Lemaire et al. [167], Zhang et al. [31], Dong and Yang [168]
2020	Kundu and Sundaresan [169]
2021	Kaliamoorthi et al. [170]
Methods based on rank and reconstruction	
2013-2014	Sainath et al. [32], Jaderberg et al. [33],
2015-2016	Lebedev et al. [171],Zhang et al. [81]
2018-2020	Lin et al. [172], [173]
Quantization methods	
2011	Vanhoucke et al. [174]
2014-2015	Denton et al. [175], Gong et al. [176], Chen et al. [38],
2016-2017	Hubara et al. [177], Rastegari et al [178], Zhou et al. [35]
2018	Jacob et al. [39], Krishnamoorthi [179], Liu et al. [180], Tung and Mori [181]
2019	Banner et al. [182], Chen et al. [183], Jung et al. [34], Zhao et al. [36]
2020	Wang et al. [184]
2021	Fan et al. [185]
Architectural design methods	
2017	Baker et al. [40], Howard et al. [186], Lin et al. [187], Zoph and Le [46]
2018	Gordon et al. [188], He et al. [109], Hsu et al. [189], Liu et al. [190], Pham et al. [191],Zhong et al. [45]
2019	Dai et al. [41],Real et al. [192], Liu et al. [43],Tan et al. [193], Zhang et al. [194]
2020	Elsen et al. [120],Evci et al. [195], Lin et al. [44]
Hybrid Methods	
2016	Chung and Shin [47], Han et al. [98]
2018-2019	Goetschalckx et al. [48], Gadosey et al. [49]
Survey Papers	
1993	Reed [11]
2018	Cheng et al. [196], [197], Lebedev and Lempitsky [50],
2019-2021	Elsken et al. [51],Menghani [52]

magnitude and sensitivity analysis methods. The survey found relatively few methods that utilise similarity and clustering, and further sub-categorization is not useful. Magnitude based methods can be sub-categorised into: (i) data dependent methods that utilise a sample of examples to assess the extent to which removing weights impacts the outputs from the next layer; (ii) data independent methods, that utilise measures such as the magnitude of a weight; and (iii) the use of optimisation methods to reduce the number of weights in a layer whilst approximating the function of the layer. Methods that utilise sensitivity analysis can be sub-categorized into those that: (i) adopt a Taylor series approximation of the loss and (ii) use sampling to estimate the change in loss when weights are removed.

The rest of this paper is organised as follows. Section II presents the background. Sections III to V describe representative methods in the three categories: magnitude based pruning, clustering and similarity, and sensitivity analysis. Section III also includes coverage of the Lottery Hypothesis, an issue about the existence of smaller networks and fine-tuning, that cuts across the different methods. Section VI presents a comparison of the published results for pruning AlexNet, ResNet and VGG to illustrate the resource provided for comparing the methods. Section VII concludes by highlighting some key insights and suggesting directions for future research.

II. BACKGROUND

This section introduces the background knowledge assumed in the survey.[1] Fig. 2 shows the structure of one of the earliest convolutional neural networks (CNNs), LeNet-5 [60], which recognises handwritten digits by applying convolutions and pooling operations to identify features. These features then provide the input to fully connected layers that classify the images. The pooling operation takes feature maps as input and reduces their size by applying an operation, such as the maximum value within a neighbourhood while the convolution operation applies filters (or kernels) to the input channels (or feature maps) to produce the output feature maps. The filters are k $\times$ k matrices that slide over the input feature maps and convolve with the corresponding elements of the input feature maps to produce the output feature maps. The elements of a filter correspond to the weights (or parameters) that are used to transform regions in feature maps in one layer to the next and need to be learned through training. The weights (or parameters), either individually or collectively as filters, are therefore the primary candidates for pruning.

The LeNet-5 model, with 60K parameters in 5 layers, achieved impressive results on a data set known as MNIST [61].[2] In a breakthrough in 2012, AlexNet built upon the concepts in LeNet-5 and developed a deeper network with over 60M parameters in 8 layers to win a competition known as ImageNet by a significant margin [62]. This success was followed by the development of architectures like VGG, ResNet, and ResNeXT that used an increasing number of layers and parameters to gain further improvements in the ImageNet competition [5]. The huge number of parameters in these models does necessitate greater computational resources and inhibits their use in embedded systems, which has motivated the research on pruning that is surveyed in this paper.

The pruning methods developed are evaluated on a range of architectures (e.g., ResNet, VGG, DenseNet) and data sets (e.g., ImageNet CIFAR, SVHN). Khan *et al.* [63] present a tutorial on deep learning architectures and Appendix A summarises the data sets. When evaluating pruning methods, the surveyed papers use the following measures to report their results:

- The Top-1 and Top-5 accuracy, which report the proportion of times the correct classification appears first or in the top 5 list of ranked results. In the sections below, unless we explicitly qualify a measure, the Top-1 accuracy should be assumed.
- The compression rate, which is the ratio of parameters before and after a model is pruned.
- The computational efficiency in terms of the FLOPS (Floating Point Operations) required to perform a classification.

The notation used in the paper is defined where it is used and also summarised in Appendix B. With this background in place, Sections III to V describe and contrast key influential studies that bring out the features of the categories of methods surveyed in this paper.

III. MAGNITUDE BASED PRUNING

This section presents pruning methods that remove weights, nodes, and filters based on a measure of magnitude or the effect filters have on the next layer. Section III-A summarizes an early influential method for pruning weights and Section III-B presents a recent hot topic, termed the Lottery Hypothesis, that reinvigorates research on the existence of smaller networks and raises issues about fine-tuning a pruned network. Section III-C describes the key ideas behind methods that prune filters and feature maps.

A. NETWORK PRUNING OF WEIGHTS

One of the first studies to utilise magnitude based pruning for deep networks is due to Han *et al.* (2015) who adopt a process in which weights below a given threshold are pruned [64].[3] Once pruned, the network is fine-tuned and the process repeated until its accuracy begins to deteriorate.

Han *et al.* [64] carry out several experiments to compare the merits of their magnitude based iterative pruning method. First, they apply their method on a fully connected network

[1] Readers unfamiliar with deep neural networks are referred to tutorial accounts such as by Goodfellow *et al.* [54] for further details

[2] Modified National Institute of Standards and Technology

[3] We use a number citation style, but include the name(date) format to highlight the date of the publication where we think it is relevant. The corresponding reference number is provided at the end of the sentence

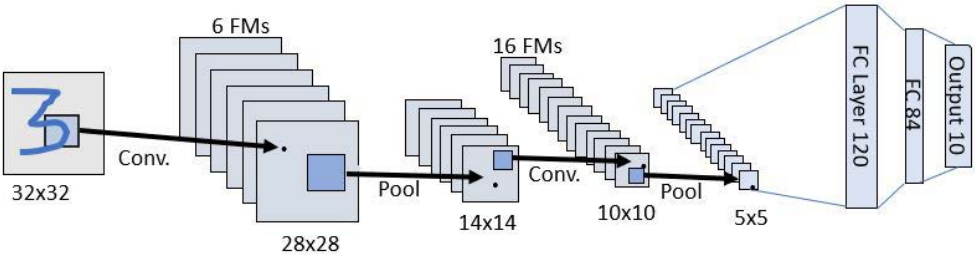

FIGURE 2. The LeNet-5 network and how it processes an input image via convolutions (Conv.) and pooling operations to produce features maps (FMs) and uses fully connected (FC) layers to perform classification.

known as LeNet-300-100 and then on Lenet-5 (Fig. 2), both of which are trained on the MNIST data. Their results show that it is possible to reduce the number of weights by a factor of 12 without compromising accuracy. Second, they apply iterative pruning to AlexNet and VGG16 trained on the ImageNet data, and show that it is possible to reduce the number of weights by a factor of 9 and 12 respectively. Thirdly, they compare the merits of using regularisation to drive down the magnitude of weights to aid subsequent pruning. They explore regularisation with both the L_1 and L_2 norms and conclude that L_1 is better immediately after pruning (without fine-tuning), but L_2 is better if the weights of the pruned model are fine-tuned. Their experiments also suggest that the earlier layers (i.e., closer to the inputs) are the most sensitive to pruning and that iterative pruning is better than pruning the required proportion of weights in one cycle (i.e., one-shot pruning).

The study by Han *et al.* [64] is notable in that (i) it demonstrated that it was possible to reduce the size of deep networks significantly without compromising accuracy, (ii) it highlighted the benefits of iterative pruning and (iii) it prompted further research on questions such as whether retraining from scratch or fine-tuning is better following pruning.

Guo *et al.* [65] note that magnitude pruning can lead to premature removal of weights that can become important given removal of other weights. To address this, they propose a method known as Dynamic Network Surgery (Dyn Surg) which maintains a mask that indicates which weights should be removed and retained in each training cycle, thereby allowing reinstatement of weights previously marked to be pruned if they turn out to be important. They compare their method with magnitude pruning, with the results showing that it is reduces the number of weights by a factor of over 17 for AlexNet on ImageNet.

B. THE LOTTERY TICKET HYPOTHESIS

One of the most interesting observations in Han *et al.* [64] is that re-initialization of the weights does not lead to accurate models and, based on their trials, it was better to fine-tune the weights of the pruned model. Following on from this observation, Frankle and Carbin (2019) propose the Lottery Ticket Hypothesis which states that: a trained network contains a subnetwork, which can be trained to be at least as accurate as the original network using no more than the number of epochs used for training the original network [66]. This subnetwork is termed a winning lottery ticket, given that it was lucky to be initialised with suitable weights.

To test this hypothesis, they propose two pruning methods. First, in a one-shot method, they use magnitude pruning to prune p% of the weights, reset the remaining weights to their initial values and retrain. Second, they utilise an iterative pruning method with n cycles, with each cycle pruning $p^{1/n}$ of the weights.

They perform experiments on the fully connected LeNet-300-100 network for the MNIST data, and variants of VGG and ResNet for the CIFAR10 data. Their experiments on the LeNet-300-100 network prune a percent of the weights from each layer except the final layer, in which the percent pruned is reduced by half. Their results with iterative pruning show that: (i) a subnetwork that is only 3.6% of its original size performs just as well, (ii) random initialization of the pruned networks results in slower learning in comparison to use of the original weight initializations, (iii) that the subnetworks (termed winning tickets) found, learn faster than the original network, (iv) there is continual improvement in the rate of learning as the size of the network reduces, but only up to a point, after which learning slows down and begins to regress to the performance of the original network, (v) iterative pruning tends to result in more accurate smaller networks than one-shot pruning.

Their experiments on the larger networks, VGG and ResNet, show that identification of winning lotteries depends on the learning rate, with a lower rate successfully identifying winning lottery subnets, and that pruning weights over all the network, as opposed to layer by layer produces better results.

These results provide good empirical evidence for the Lottery Hypothesis and the award of a best paper prize in the 2019 International Conference on Learning Representations is indicative of the significance of the paper and the attention it has attracted.

In their paper, ''Rethinking the value of network pruning'', Liu *et al.* (2019) challenge the claim that it is better to utilise the initial weights of a pruned model when compared with random initialization [67]. To test this, they carry out experiments on VGG, ResNet, and DenseNet using the CIFAR10, CIFAR100, and ImageNet data. They define three types of

pruning regime: structured pruning, where the proportion of channels that are pruned per layer is predefined; automatic pruning, where the proportion of channels pruned overall is predefined but the per layer rate is determined by the algorithm; and unstructured weight pruning, where only the proportion of weights pruned is predefined. Their results suggest that for structured and automatic pruning, random initialization is equally (if not more) effective. However, for unstructured networks, random initialization can achieve similar results on small data sets but for large scale data such as ImageNet, fine-tuning produces better results.

At first sight, their findings contradict the Lottery Hypothesis. However, in a follow up study, Frankle *et al.* (2019) acknowledge that setting the weights of pruned networks to their initial values does not work well on larger networks and suggest that methods for retraining from random initializations do not work well either, except for moderate levels of pruning (up to 30%) [68]. They therefore propose setting the weights to those obtained in a later iteration of training, which they then demonstrate to be beneficial in identifying good initialization of weights for larger scale problems such as ImageNet.

The above studies focus on empirical evaluations of networks trained and used on the same data sets, and primarily on image processing classification tasks. Morcos *et al.* (2019) explore a number of other interesting questions [69]:

- Are the lotteries found for one image classification task transferable to other tasks?
- Are lotteries observable in other tasks (such as natural language processing), and architectures?
- Are they transferable across different optimizers?

To explore these questions, they carry out experiments with VGG19 and ResNet50 using six data sets (Fashion-MNIST, SVHN, CIFAR10, CIFAR100, ImageNet, Places365), in which the lotteries (i.e., subnetworks with initializations) identified for one task are used for another task. Their experiments use iterative magnitude based pruning, selecting 20% of the weights over all the layers, and with late setting of weights (as proposed in Frankle *et al.* [66]). The results are interesting: in general, winning initializations carry across similar image processing tasks and winning tickets from larger scale tasks were more transferable than the tickets from the smaller scale tasks. In some cases, for example, the use of VGG19 on the Fashion-MNIST data, the winning tickets obtained from the use of VGG19 on the larger data sets (CIFAR100, ImageNet) performed better than those obtained directly from the Fashion-MNIST data.

Hubens *et al.* (2020) carry out empirical trials that confirm similar results on the size of the pruned networks [70]. They show that when a network is trained on a larger data set, such as ImageNet, and transferred and fine-tuned for a different task, pruning can result in a smaller network than if it was trained from scratch on the new task.

Morcos *et al.* (2019) carry out experiments in which lottery tickets are identified using one optimizer, ADAM (adaptive moment estimation), and then utilise a different optimizer, SGD (Stochastic Gradient Descent) with momentum, and vice versa on the CIFAR10 data. Their results suggest that, in general, winning tickets are optimizer independent [69].

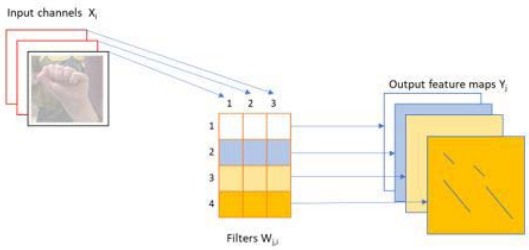

FIGURE 3. Illustration of how the feature maps are computed, where $W_{j,i}$ are the k × k filters used on the input channels X_i to obtain output feature maps Y_j.

To test if the lottery hypothesis holds in other types of problems, Yu *et al.* (2019) carry out experiments on natural language processing (NLP) and control tasks in games [71]. For NLP, they utilise LSTMs for the Wikitext-2 data [72] and Transformer models for translating news in English to German [73]. The experiments were carried out with 20 rounds of iterative pruning and with one-shot pruning. A pruning rate of 20% was used and following pruning, weights were reset to those learned during a later round of training. For control tasks, they utilise Reinforcement Learning (RL) and carry out experiments on fully connected networks used for 3 OpenAI Gym environments [74] and 9 Atari games that utilise convolutional networks [75].

From their results on NLP and the RL control tasks, they conclude that both iterative pruning and late setting of weights are superior in comparison to random initialization of pruned networks, with iterative pruning being essential when a significant number of weights (i.e., more than two-thirds) are pruned. For the Atari games, the results varied: in one case, it led to improvements over the original network (Berzerk game) while in another, an initial improvement was followed by a significant drop in accuracy as the amount of pruning increased (Space Invaders game). In other cases, pruning resulted in a reduction in performance (e.g., Assault game). Thus in summary, Yu *et al.* [71] provide some evidence that the lottery hypothesis holds for NLP tasks and for some control tasks that utilise RL.

C. PRUNING FEATURE MAPS AND FILTERS

Although the kind of methods described in Section III-A result in fewer weights, they require specialist libraries or hardware for processing the resulting sparse weight matrices [76]–[78]. In contrast, pruning at higher levels of granularity, such as pruning filters and channels benefits from the optimizations already available in many current toolkits. This has led to a number of methods for pruning feature maps and filters which are summarized in this section.

To appreciate the intuition and notation behind these methods, it is worth bearing in mind how filters are applied to the input channels to produce the output feature maps. Fig. 3

illustrates the process, showing how an image with 3 channels is taken as input and convolved with the filters to produce the 4 output feature maps. Given the visualisation offered by Fig. 3, how can one best prune the filters and channels? The survey revealed three main directions of research:

- **Data dependent channel pruning methods**, which are based on the view that when different inputs are presented, the output channels (i.e., feature maps) should vary given they are meant to detect discriminative features.
- **Data independent pruning methods**, that use properties of the filters and output channels, such as the proportion of zeros present, to decide which filters and channels should be pruned.
- **Optimization based channel approximation pruning methods**, that use optimization methods to recreate the filters to approximate the output feature maps.

The following describes and contrasts methods that typify these three directions.

1) PRUNING BASED ON VARIANCE OF CHANNELS AND FILTERS

Polyak & Wolf (2015) propose two methods for pruning channels: Inbound pruning, which aims to reduce the number of channels incoming to a filter and Reduce and Reuse pruning, which aims to reduce the number of output channels [79].

The idea behind Inbound pruning is to assess the extent to which an input channel's contribution to producing an output feature map varies with different examples. This assessment is done by applying the network to a sample of the images and then using the variance in a feature map as a measure of its contribution.

More formally, given $W_{j,i}$, the j_{th} filter for the i^{th} input channel, and X_i^p, the input from the i^{th} channel for the p^{th} example, the contribution to the j^{th} output feature map, $Y_{j,i}^p$ is defined by:

$$Y_{j,i}^p = \left\| W_{j,i} \cdot X_i^p \right\|_F \tag{1}$$

Given this definition, the measure used to assess the variation in its contribution, $\sigma_{j,i}^2$ from the N samples is:

$$\sigma_{j,i}^2 = var\left(\left\{ Y_{j,i}^p \mid p = 1 \ldots N \right\}\right) \tag{2}$$

Inbound pruning uses this measure to rank the filters $W_{j,i}$ and removes any that fall below a specified threshold.

The Reduce and Reuse pruning method focuses on assessing the variations in the output feature maps when different samples are presented. That is, the method first computes the variations in the output feature maps σ_j^2 using:

$$\sigma_j^2 = var\left(\left\| \sum_{i=1}^{m} Y_{j,i}^p \right\|_F \mid p = 1 \ldots N\right) \tag{3}$$

where m is the number of channels and N is the number of samples. Reduce and Reuse then uses this measure to retain a proportion of the output feature maps and corresponding filters that results in the greatest variation.

Removal of an output feature map is problematic given it is expected as an input channel in the next layer. To overcome this, they approximate a removed channel using the other channels. That is, if Y_i, Y_i' are the outputs of a layer before and after pruning a layer respectively, the aim is to find a matrix A such that:

$$\min_A \sum_i \left\| Y_i - AY_i' \right\|_2^2 \tag{4}$$

The matrix A is then included as an additional convolutional layer of 1×1 filters along the lines proposed by Lin *et al.* [80].

Polyak & Wolf [79] evaluate the above approach on the Scratch network, using the CASIA-WebFace and the Labeled Faces in the Wild (LFW) data sets. They utilise layer by layer pruning, where each layer is pruned, and the network fine-tuned before moving on to the next layer. They experiment with their two pruning methods individually and in combination, and compare the results with the use of random pruning, a low rank approximation method [81] and Fitnets, a method that uses the Knowledge Distillation approach to learn smaller networks [82]. In the experiments with the Inbound pruning method, they prune channels where $\sigma_{j,i}^2$ is below a given threshold, selected such that the overall accuracy is maintained above 84%. For the experiments with the Reduce and Reuse method, they try different levels of pruning: 50%, 75%, and 90% for the earlier layers followed by 50% for the later layers. The adoption of a lower pruning rate for the later layers follows an observation that heavy pruning of the later layers results in a marked reduction in accuracy.

The results from their experiments show that: (i) the variance based method is more effective than use of random pruning, (ii) the use of fine-tuning does help in recovering accuracy, especially in the later layers, (iii) their methods result in greater compression than use of a low rank method and the use of Fitnets when applied to the Scratch network.

2) ENTROPY-BASED CHANNEL PRUNING

Instead of the variance, Luo & Wu (2017) propose an entropy-based metric to evaluate the importance of each filter [83]. In their filter pruning method, if a feature map contains less information, its corresponding filter is considered less important, and could be pruned. To compute the entropy of a particular feature map, they first sample the data and obtain a set of feature maps for each filter. Each feature map is reduced to a point measure using a global average pooling method, and the set of measures associated with each filter are discretized into q groups. The entropy of a filter, H_j is then used to assess the discriminative power of a filter [83]:

$$H_j = \sum_{i=1}^{q} P_i \log\left(P_i\right) \tag{5}$$

where P_i is the probability of an example being in group i.

They explore both one-shot pruning followed by fine-tuning and layer wise pruning in which they fine-tune with just one or two epochs of learning immediately after pruning a layer. Their layer wise strategy is an interesting compromise between fully fine-tuning after pruning each layer, which can be computationally expensive, and only fine-tuning at the end, which can fail to take account of the knock-on effects of pruning previous layers.

They evaluate the merits of the entropy-based method by applying it to VGG16 and ResNet-50 on the ImageNet data. For VGG16, they focus on the first 10 layers and, also replace the fully connected layers by use of average pooling to obtain further reductions. They compare their results on VGG16 with those obtained by the magnitude based pruning method and APoZ method (described below). Their results suggest that: (i) the entropy-based method achieves more than a 16 fold compression, though this is at the expense of a 1.56% reduction in accuracy, (ii) use of magnitude pruning results in a 13 fold compression, and (iii) APoZ results in a 2.7 fold compression. However, it should be noted that the higher compression rate achieved by the use of entropy includes the reduction due to the replacement of the fully connected layers by average pooling, without which the use of the entropy-based method leads to a lower compression rate than APoZ (Table 3 in [83]).

3) APoZ: NETWORK TRIMMING BASED ON ZEROS IN A CHANNEL

In contrast to the use of samples of data to compute the variance of a feature map or its entropy, Hu *et al.* (2016), suggest a direct method that is based on the view that the number of zeros in an output feature map is indicative of its redundancy [84]. Based on this view, they propose a method that uses the average number of zero activations (APoZ) in a feature map as a measure of the weakness of a filter that generates the feature map.

Their experiments are with LeNet5 on MNIST and VGG16 on ImageNet and aimed at first finding the most appropriate layers to prune and then to iteratively prune these layers in a bespoke way that maintains or improves accuracy. Following pruning, they experiment with both retraining from scratch and fine-tuning the weights and prefer the latter given better results.

For LeNet-5, they observe that most of the parameters (over 90%) are in the 2nd convolution layer and the first fully connected layer and hence they focus on pruning these two layers in four iterations of pruning and fine-tuning, resulting in the size of the convolutional layer reducing from 50 to 24 filters and the number of neurons in the fully connected layer reducing from 500 to 252. Overall, this represents a compression rate of 3.85.

For VGG16, they also focus on one convolutional layer that has 512 filters and a fully connected layer with 4096 nodes. After 6 iterations, they reduce these to 390 filters and 1513 nodes, achieving a compression rate of 2.59.

4) PRUNING SMALL FILTERS AND FILTER SKETCHING

Li *et al.* (2017) extend the idea of magnitude pruning of weights to filters by proposing the removal of filters that have the smallest absolute sum among the filters in a layer [76]. That is, if the filters for producing the j^{th} feature map are $W_{j,i} \in \mathbb{R}^{k \times k}$ and m is the number of input feature maps, then the magnitude of the j^{th} filter is defined by:

$$s_j = \sum_{i=1}^{m} \left\| W_{j,i} \right\|_1 \tag{6}$$

Once the s_j are computed, a proportion of the smallest filters together with their associated feature maps and filters in the next layer are removed. After a layer is pruned, the network is fine-tuned, and pruning is continued layer by layer.

To test this approach, they carry out experiments on VGG16 and ResNet56 & 110 on CIFAR10 and ResNet34 on ImageNet. By analyzing the sensitivity of the layers through experimentation, they determine appropriate pruning ratios for each layer that would not compromise accuracy significantly. Overall, for VGG16, they are able to prune the parameters by 64%. A significant proportion of this pruning is in layers 8 to 13 which consist of the smaller filters (2×2 and 4×4), which they notice can be pruned by 50% without reducing accuracy. The level of pruning for the other networks is more modest, with the best pruning rate for ResNet-56 and ResNet110 on CIFAR10 being 3.7% and 32.4% respectively, and for ResNet-34 on ImageNet being 10.8%.

They also compare their approach with the variance-based method described above and conclude that use of the above measure over filters performs at least as well but without the additional need to compute the feature maps via samples of the data.

A more recent method, proposed by Lin *et al.* (2020), known as filter sketch also aims to reduce the number of filters without the need to sample examples [85]. The key idea in filter sketching is to minimize the difference between the co-variances of the original set of filters and the reduced set. Although this can be done using optimization methods, filter sketch utilises a greedy algorithm known as Frequent Direction [86] which is more efficient.

Lin *et al.* [85] evaluate the filter sketch method on GoogleNet, ResNet56 and ResNet110 using the CIFAR10 data, and on ResNet50 with the ImageNet data. The results show that it performs well relative to the method for pruning small filters and a method that uses optimization to prune channels (described below in Section III-C7) in terms of reducing the number of parameters without a significant loss in accuracy.

5) PRUNING FILTERS BASED ON GEOMETRIC MEDIAN

He *et al.* (2019) point out that pruning based on the magnitude of filters assumes that there are some small filters and that the spread of magnitude is wide enough to adequately distinguish those filters that contribute from those do not contribute [87]. So, for example, if most of the weights are small, one could

end up removing a significant number of filters and if most of the filters have large values, no filters would be removed, even though there may be filters that are relatively small. Hence, they propose a method based on the view that the geometric median of the filters shares most of the information common in the other filters and hence a filter that is close to it can be covered by the other filters if deleted. Computing the geometric median can be time-consuming, so they approximate its computation by assuming that one of the filters will be the geometric mean. Their pruning strategy is to prune and fine-tune repeatedly using a fixed pruning factor for all layers.

They carry out an evaluation with respect to several methods including pruning small filters [76], ThiNet [88], Soft filter pruning [89], and NISP [90]. These methods are evaluated on ResNets trained on the CIFAR10 and ImageNet data, with pruning rates of 30 and 40 percent. In general, the drop in accuracy is similar across the different methods, though there is a significant reduction in FLOPS when using the geometric median method on ResNet-50 (53.5%) compared to the other methods (e.g., ThiNet 36.7%, Soft filter pruning 41%, NISP 44%).

6) ThiNet AND AOFP

Luo *et al.* (2017) formulate the pruning task as an optimization problem and propose a system ThiNet in which the objective is to find a subset of input channels that can best approximate the output feature maps [88]. The channels not in the subset and their corresponding filters can then be removed. Solving the optimization problem is computationally challenging, so ThiNet uses a greedy algorithm that finds a channel that contributes the least, adds it to the list to be removed, and repeats the process with the remaining channels until the number of channels selected equals the number to be pruned. Once a subset of filters to be retained is identified, their weights are obtained by using least squares to find the filters W that minimize [88]:

$$\sum_{i=1}^{m} \left(Y_i - W^T \cdot X_i \right)^2 \tag{7}$$

where Y_i are the m sampled points in the output channels and X_i their corresponding input channels.

They evaluate their approach in two sets of experiments. In the first, they adapt VGG16, replacing the fully connected layers by global average pooling (GAP) layers, apply it to the UCSD-Birds data and then prune it using ThiNet, APoZ and the small filters method. Their results show there is less degradation in accuracy with ThiNet than ApoZ, which in turn, is better than the small filters method.

In their second set of experiments, they utilise VGG16 and ResNet50 trained on the ImageNet data. For VGG16, their procedure involves pruning a layer and then minor fine-tuning with one epoch of training with an additional epoch at the end of each group of convolutional layers and a further 12 epochs of fine-tuning after the final layer. With the use of GAP, ThiNet, reduces the number of parameters by about 94% at the expense of a 1% reduction in Top-1 accuracy. For ResNet, ThiNet is applied on the first two convolutional layers of each residual block, keeping the output dimensions of the blocks the same. After pruning each layer, one epoch of fine-tuning is performed, and 9 epochs are used for fine-tuning at the end. The results show that ThiNet is able to halve the number of parameters with a 1.87% loss in Top-1 accuracy.

Ding *et al.* (2019) propose a similar method to ThiNet, called Approximated Oracle Filter Pruning (AOFP), which aims to identify the subset of filters, which if removed, will have the least effect on the feature maps in the next layer [91]. However, whereas, the search procedure adopted in ThiNet uses a greedy bottom up approach, AOFP adopts a top-down binary search in which half of the filters in a layer are randomly selected and set to be pruned. The effect of removing these filters on the feature map produced in the next layer is measured and recorded against each filter that is set as pruned. This process is repeated for different random selections, and the average effect per filter used as an indication of the effect of removing a filter. The top 50% of the filters that would result in the worst effect if removed are retained and the process repeated unless this would result in an unacceptable reduction in accuracy. In comparison to ThiNet, and other methods, AOFP does not require the rate of pruning to be fixed in advance of pruning a layer.

AOFP is evaluated by pruning AlexNet, VGG and ResNet trained on the CIFAR10 and ImageNet data. They compare AOFP with several methods including: ThinNet, Network Slimming [92], Pruning using Agents [93], Online Filter Weakening [15], NISP [90], Optimizing Channel Pruning [94], Structured Probabilistic Pruning [95], Autopruner [96], and ISTA [97], with their results showing that AOFP is capable of greater reductions in FLOPS without compromising accuracy.

7) OPTIMIZING CHANNEL SELECTION WITH LASSO REGRESSION

He *et al.* (2017) also formulate channel selection as an optimization problem [94]. Given a channel Y obtained by applying a filter W_i to m input channels X_i:

$$Y = \sum_{i=1}^{m} X_i W_i^T \tag{8}$$

They define the task as one to optimize:

$$\begin{aligned} &\arg\min_{\beta, W} \frac{1}{2} \left\| Y - \sum_{i=1}^{c} \beta_i X_i W_i^T \right\|_F^2 \\ &\text{subject to } \|\beta\|_0 \leq p \end{aligned} \tag{9}$$

where p indicates the number of channels retained and $\beta_i \in \{0, 1\}$ indicates the retention or removal of a channel.

In contrast to ThiNet, which adopts a greedy heuristic to solve this optimization problem, He *et al.* (2017) relax the problem from L_0 to L_1 regularization and utilise LASSO

regression to solve [94]:

$$\arg\min_{\beta,W} \frac{1}{2}\left\| Y - \sum_{i=1}^{c} \beta_i X_i W_i^T \right\|_F^2 + \lambda\|\beta\|_1$$
$$\text{subject to } \|\beta\|_0 \leq p \tag{10}$$

Following the selection of the channels they utilise least squares to obtain the revised weights in a manner similar to the approach adopted in ThiNet.

They carry out empirical evaluations on VGG16, ResNet50 and a version of the Xception network, trained on the CIFAR10 and ImageNet data. They also explore the extent to which the pruned models can be used for transfer learning by using them for the PASCAL VOC 2007 object detection task.

In their first set of experiments, they evaluate their method on single layers of VGG16 trained on CIFAR10 without any fine-tuning, and show that their algorithm maintains Top-5 accuracy better than the method of pruning small filters. They also include results from a naïve method that selects the first k feature maps and show that, for some layers (e.g. conv3_3 in VGG16), this sometimes performs better than the method of pruning small filters, highlighting a potential weakness of magnitude-based pruning.

In a second set of experiments, with VGG16 on CIFAR10, they apply their method on the full network, using bespoke pruning ratios for the layers and fine-tuning to achieve 2, 4 and 5 fold improvements in run-time, but resulting in drops of Top-5 accuracy of 0%,1%, and 1.7% respectively. In comparison, the method for pruning small filters results in larger drops of 0.8%, 8.6% and 14.6%.

Their experiments on ResNet50 adopt bespoke pruning rates per layer, retaining 70% of layers that are very sensitive to pruning, and 30% of the less sensitive layers. The Top-5 accuracy results on ImageNet show a two-fold improvement in run-time at the expense of a 1.4% drop in accuracy compared to a baseline accuracy of 92.2%, while the results on the Xception network show a drop of 1% in accuracy from a baseline of 92.8%.

The experiment on using a pruned version of a VGG16 model on the PASCAL VOC 2007 object detection benchmark task results in a 2-fold increase in speed with a 0.4% drop in average precision.

IV. PRUNING BASED ON SIMILARITY AND CLUSTERING

Given that neural networks can be over-parametrised, it is plausible that there could be duplicate weights or filters that perform similar functions and can be removed without impacting accuracy [19], [20], [98]–[100].

RoyChoudry *et al.* (2017) explore this hypothesis by using the inner product of two filters (or weight matrices) as a measure of similarity [19]. Their pruning algorithm involves grouping filters that are similar and then replacing each group of filters by their mean filter. They carry out experiments with both a multilayer perceptron (MLP) and a CNN for the CIFAR10 data. The MLP has three layers: the first two are fully connected layers and the third is a softmax layer with 10 nodes representing the class for CIFAR10. The CNN has two convolution layers, each followed by a ReLU, and a 2 × 2 max pooling layer. The convolutional layers are followed by two fully connected layers to perform the classification. In both cases, the first layer is varied with 100, 500 and 1000 units (nodes or filters) to explore the effects of increasing over parametrisation. Their main finding is that there is a much greater propensity for similar weights/filters to occur in MLPs than in CNNs. As a consequence, there is a greater opportunity for using similarity as a basis for pruning MLPs than for pruning CNNs. Nevertheless, their results suggest that a similarity based pruning algorithm is better at retaining accuracy than using the small filters method.

Ayinde *et al.* (2019) also develop a method that uses clustering to identify similar filters [78]. They too adopt the inner product as a measure of similarity, but use an agglomerative hierarchical clustering method to group similar filters and replace the filters by randomly selecting one filter from each cluster. They carry out various experiments with VGG16 on CIFAR10 and ResNet34 on ImageNet. For the trial on VGG16 with the CIFAR10 data, they show that, once an optimal value for the threshold for similarity is determined, their method achieves both a better pruning rate and accuracy than other methods, including pruning of small filters, Network Slimming [92], a method that uses regularization to identify weak channels, and try-and-learn [93], a method that uses sensitivity analysis.

V. SENSITIVITY ANALYSIS METHODS

The primary goal of pruning is to remove weights, filters and channels that have least effect on the accuracy of a model. The magnitude and similarity based methods described above address this goal implicitly by using properties of weights, filters and channels that can affect accuracy. In contrast, this section presents methods that use sensitivity analysis to model the effect of perturbing and removing weights, filters and channels on the loss function.

Section V-A describes methods that assess the importance of channels and Sections V-B to V-D present the development of a line of research that approximates the effect of perturbing the weights on the loss function using the Taylor Series, from the earliest work which developed methods for MLPs to the more recent research on methods for pruning CNNs.

A. PRUNING BY ASSESSING THE IMPORTANCE OF NODES AND CHANNELS

Skeletonization, a method proposed by Mozer & Smolensky(1988), was one of the earliest approaches to pruning neural networks [9]. To calculate the effect of removing nodes, Skeletonization introduced the notion of attentional strength to denote the importance of nodes when computing activations. Given the attentional strengths of the nodes, α_i,

the output y_j from node j, is defined by:

$$y_j = f\left(\sum_i w_{ji}\alpha_i y_i\right) \tag{11}$$

where f is assumed to be the sigmoid function. The importance of a node ρ_i is then defined in terms of the difference in loss when α_i is set to zero and when it is set to one and can be approximated by the derivative of the loss with respect to the attentional strength α_i:

$$\rho_i = \mathcal{L}_{\alpha_i=0} - \mathcal{L}_{\alpha_i=1} \approx -\left.\frac{\partial \mathcal{L}}{\partial \alpha_i}\right|_{\alpha_i=1} \tag{12}$$

Through experimentation, they found that the linear loss worked better a than the quadratic loss because the difference between the outputs and targets was small following training. In addition, they noticed that the $\partial\mathcal{L}(t)/\partial\alpha_i$ were not stable with time, so they used a weighted average measure to compure the importance $\widehat{\rho_i}$:

$$\widehat{\rho_i}(\mathrm{t}+1) = 0.8\widehat{\rho_i}(\mathrm{t}) + 0.2\frac{\partial \mathcal{L}(t)}{\partial \alpha_i} \tag{13}$$

Mozer & Smolensky [9] present a number of small but very interesting experiments. These include generating examples where the output is correlated to four inputs, A,B,C, and D, with full correlation on A and reducing to no correlation with D. They provide this as input to a network with one hidden node and following training they observe that the weights from the inputs to the hidden node follow the correlations, although the relevance measure only shows input node A as important, providing some reassurance that the measure is different from the weights. In another example, they develop a network to model a 4-bit multiplexor network, which has 4 bits as input and two bits to control which of the 4 bits is output. They try two network configurations: in the first, they utilise 4 hidden nodes and in the second they utilise 8 hidden nodes and use skeletonization to reduce its size to 4 hidden nodes. When limiting training to 1000 epochs, they find that starting with 4 hidden nodes initially, results in failure to converge in 17% of the cases, while beginning with 8 hidden layer nodes followed by skeletonization converges in all the cases and, also retains accuracy. This appears to be one of the first demonstrations that, to begin, it may be necessary to overparameterize a network in order to find winning lotteries.

This idea of assessing the importance of nodes has been extended to channels by two methods, namely Network Slimming [92] and Sparse Structure Selection (SSS) [55], that learn a measure of importance as part of the training process. Both utilise a parameter γ for each channel (analogous to the attentional strength) which scales the output of a channel. Given a loss function $\mathcal{L}$, the new loss $\mathcal{L}'$ is defined with an additional regularization term over the scaling factors γ:

$$\mathcal{L}' = \mathcal{L} + \lambda \sum_u g(\gamma) \tag{14}$$

where the function g is selected as the L_1 norm to reduce γ towards zero (as in Lasso regression).

The two methods differ in the way they implement the training process aimed at minimizing $\mathcal{L}'$ with Network Slimming taking advantage of the use of batch normalization layers that are sometimes present following convolutional layers while SSS implements a more general process that does not assume the presence of batch layers, and allows use of scaling factors for blocks (such as residual and inception blocks) that can enable reduction of the depth of a network.

Huang and Wang(2018) experiment with SSS on the CIFAR-10, CIFAR-100, and ImageNet data on VGG16, and ResNet [55]. For CIFAR10, SSS is able to reduce the number of parameters in VGG16 by 30% without loss of accuracy. For ResNet-164, it is able to achieve a 2.5 times speedup at the cost of a 2% loss in accuracy for CIFAR-10 and CIFAR-100.

For VGG16 on ImageNet, SSS is able to reduce the FLOPs by about 75%, though parameter reduction is minimal, which is consistent with other methods given the large number of parameters in the fully connected layers in VGG16. On ResNet50, SSS achieves a 15% reduction in FLOPs at a cost of a 0.1% reduction in Top-1 accuracy.

B. PRUNING WEIGHTS WITH OBD AND OBS

Several studies utilise the Taylor series to approximate the effect of weight perturbations on the loss function [22], [23], [101]. Given the change in weights ΔW, a Taylor Series approximation of the change in loss $\Delta\mathcal{L}$ can be stated as [102]:

$$\Delta\mathcal{L} = \frac{\partial \mathcal{L}^T}{\partial W}\Delta W + \frac{1}{2}\Delta W^T H \Delta W + O\left(\|\delta W\|^3\right) \tag{15}$$

where H is a Hessian matrix whose elements are the second order derivatives of the loss with respect to the weights:

$$H_{ij} = \frac{\partial^2 \mathcal{L}}{\partial w_i \partial w_j} \tag{16}$$

Most methods that adopt this approximation assume that the third order term is negligible. In Optimal Brain Damage (OBD), LeCun *et al.* (1990), also assume that the first order term can be ignored given that the network will have been trained to achieve a local minima, resulting in a simplified quadratic approximation [22]:

$$\Delta\mathcal{L} = \frac{1}{2}\Delta W^T H \Delta W \tag{17}$$

Given the large number of weights, computing the Hessian is computationally expensive, so they also assume that the change in loss can be approximated by the diagonal elements of the Hessian, resulting in the following measure of the saliency s_k of a weight w_k:

$$s_k = \frac{H_{kk} w_k^2}{2} \tag{18}$$

where the second order derivatives, H_{kk} are computed in a manner similar to the way the gradient is computed in backpropagation.

Hassibi *et al.* (1993) argue that ignoring the non-diagonal elements of a Hessian is a strong assumption, and propose

an alternative pruning method, called Optimal Brain Surgeon (OBS), that aims to take account of all the elements of a Hessian [23], [24].

Using a unit vector, e_m, to denote the selection of the m_{th} weight as the one to be pruned, OBS reformulates pruning as a constraint-based optimization task:

$$\min_{\delta w} \left\{ \frac{1}{2} \delta w^T \cdot H \cdot \delta w \right\} \quad \text{subject to } e_m^T . \delta w + \delta w_m = 0 \tag{19}$$

Formulating this with a Lagrangian multiplier, λ, the task is to minimize:

$$\frac{1}{2} \delta w^T \cdot H \cdot \delta w + \lambda \left(e_m^T \cdot \delta w + \delta w_m \right) \tag{20}$$

By taking derivatives and utilizing the above constraint, they show the saliency, s_k of weight w_k can be computed using:

$$s_k = \frac{1}{2} \frac{w_k^2}{\left[H^{-1} \right]_{k,k}} \tag{21}$$

They show that on the XOR problem, modelled using a MLP network with 2 inputs, 2 hidden layer nodes and one output, OBS is better at detecting the correct weights to delete than OBD or magnitude pruning. They also show that OBS is able to significantly reduce the number of weights required for neural networks trained on the Monk problems [103] and for NetTalk [104], one of the classical applications of neural networks, it is able to reduce the number of weights required from 18000 to 1560.

C. PRUNING FEATURE MAPS WITH FIRST-ORDER TAYLOR APPROXIMATIONS

The methods described in Section V-B focus on the effect of removing weights in a fully connected network. Molchanov *et al.* (2016) introduce a method that uses the Taylor series to approximate what happens if a feature map is removed [105]. In contrast to OBD and OBS, which assume that the first order term can be ignored, they adopt a first order approximation, ignoring the higher order terms, primarily on grounds of computational complexity. Using a first order approximation seems odd given the convincing argument for ignoring these terms; however they argue that although the first order gradient tends to zero, the expected value of the change in loss is proportional to the variance, which is not zero and is a measure of the stability as a local solution is reached. Given a feature map with N elements $Y_{i,j}$, the first order approximation using the Taylor series leads to the following measure of the absolute change in loss [105]:

$$\Delta \mathcal{L}(Y) = \left| \frac{1}{N} \sum_{i,j} \frac{\partial \mathcal{L}}{\partial Y_{i,j}} Y_{i,j} \right| \tag{22}$$

The scale of this measure will vary in different layers, and they therefore apply L_2 normalization within each layer. The pruning process they adopt involves selecting a feature map using the measure, pruning it, and then fine-tuning the network before repeating the process until a stopping condition, that takes account of the need to reduce the number of FLOPs while maintaining accuracy, is met. Their experiments reveal several interesting findings:

1) From experiments on VGG16 and AlexNet on the UCSD-Birds and Oxford-Flowers data, they show that the features maps selected by their criteria correlate significantly more closely to those selected by an oracle method than OBD and APoZ. On the ImageNet data, they find that OBD correlates best when AlexNet is used.
2) In experiments on transfer learning, where they fine-tune VGG16 on the UCSD-Birds data, they present results showing that their method performs better than APoZ and OBD as the number of parameters pruned increases. In an experiment in which AlexNet is fine-tuned for the Oxford Flowers data, they show that both their method and OBD perform better than APoZ.
3) In a striking example of the potential benefits of pruning, they demonstrate their method on a network for recognizing hand gestures that requires over 37 GFLOPs for a single inference but only requires 3 GFLOPs after pruning, all be it with a 2.6% reduction in accuracy.

In a follow up publication, Molchanov *et al.* (2019) acknowledge some limitations of the above approach, namely that assuming that all layers have the same importance does not work for skip connections (used in the ResNet architecture) and that assessing the impact of changes in feature maps leads to increases in memory requirements [106]. They therefore propose an alternative formulation, also using a Taylor series approximation, but based on estimating the squared loss due to the removal of the m_{th} parameter:

$$(\Delta \mathcal{L}_m)^2 = \left(g_m w_m + \frac{1}{2} w_m H_m W \right)^2 \tag{23}$$

where g_m is the first order gradient and H_m is the m^{th} row of the Hessian matrix. The measure of importance of a filter is then obtained by summing the contributions due to each parameter in a filter.

The pruning algorithm employed proceeds as follows. In each epoch, they utilise a fixed number of mini-batches to estimate the importance of each filter and then, based on their importance, a predefined number of filters is removed. The network is then fine-tuned, and the process repeated until a pruning goal, such as the desired number of filters or a limit for an acceptable drop in accuracy is reached.

They carry out initial experiments on versions of LeNet and ResNet on the CIFAR10 data, using both the second and first order approximations (in equation 23) and given that the results from both correlate well with an oracle method, they utilise the first-order measure which is significantly more efficient to compute.

In experiments with versions of ResNet, VGG, and DenseNet on the ImageNet data, they consider the effect of using the measure of importance at points before and after the batch normalization layers, and conclude that the latter option results in greater correlation with an oracle method. The results from their method show that it works especially well on pruning ResNet-50 and ResNet-34, outperforming the results from ThiNet and NISP. The reported results for other networks are also impressive, with their method able to prune 76% of the parameters in VGG with a 0.19% loss in accuracy and able to reduce the number of parameters in DenseNet by 43% at the expense of a 0.29% reduction in accuracy.

D. PRUNING FEATURE MAPS WITH SECOND-ORDER TAYLOR APPROXIMATIONS

The first order methods described above assume minimal interaction across channels and filters. This section summarizes recent pruning methods that aim to take account of the effect of the potential dependencies amongst the channels and filters.

In a method called EigenDamage, that also utilises the Taylor series approximation, Wang *et al.* (2019) revisit the assumptions made by OBD and OBS when approximating the Hessian [101]. To motivate their method, they begin by illustrating that although OBS is better than OBD when pruning one weight at a time, it is not necessarily superior when pruning multiple weights at a time. This is primarily because OBS does not correctly model the effect of removing multiple weights, especially when they are correlated. To avoid this problem, they utilise a Fisher Information Matrix to approximate the Hessian and then they utilise a method, proposed by Grosse & Martens [107], to represent a Fisher Matrix by a Kronecker Factored Eigenbasis (KFE). This reparameterization allows pruning to be done in a new space in which the Fisher Matrix is approximately diagonal. Pruning can thus be done by first mapping the weights to a KFE space in which they are approximately independent, and then mapping back the results to the original space.

EigenDamage is evaluated on VGG, and ResNet on the CIFAR10, CIFAR100 and the Tiny-ImageNet data. Experiments are carried out with one-shot pruning where fine-tuning is performed at the end and with iterative pruning in which fine-tuning is performed after each cycle. In both cases, the results show that EigenDamage outperforms adapted versions of OBD, OBS and Network Slimming.

Peng *et al.* (2019) also utilise a Taylor series approximation to develop a Collaborative Channel Pruning (CCP) method that is based on a measure of the impact of a combination of channels [108]. Given a mask β, where $\beta_i = 1$, indicates the retention of a channel and $\beta_i = 0$, indicates a channel to be pruned, they formulate the task as one to find the β_i that minimize the loss $\mathcal{L}$:

$$\mathcal{L}(\beta, W) = \mathcal{L}(W) + \sum_{i=1}^{c_o} (\beta_i - 1)\, g_i^T w_i + \frac{1}{2} \sum_{i=1,j=1}^{c_o} (\beta_i - 1)\left(\beta_j - 1\right) w_i^T H_{i,j} w_j \quad (24)$$

where g_i are the first order derivatives of the loss with respect to the weights in the i^{th} output channel, $H_{i,j}$ are Hessians, and c_o denotes the number of output channels.

By setting $u_i = g_i^T w_i$ and $s_{i,j} = \frac{1}{2} w_i^T H_{i,j} w_j$ the above equation can be written as the following 0-1 quadratic optimization problem [108]:

$$\min_{\beta_i} \sum_{i=1}^{c_o} u_i\,(\beta_i - 1) + \sum_{i=1,j=1}^{c_o} s_{i,j}\,(\beta_i - 1)\left(\beta_j - 1\right) \quad \text{subject to: } \|\beta\|_0 = p \text{ and } \beta_i \in \{0, 1\} \quad (25)$$

where p denotes the number of channels to be retained in a layer. They note that the gradients g_i and hence u_i can be computed in linear time. However, given the complexity of computing the Hessian matrices, they derive first order approximations for the loss functions, which they adopt when computing $s_{i,j}$. To solve the quadratic optimization problem, they relax the constraint to $\beta_i \in [0, 1]$ and use a quadratic programming method to find the β_i which are used to select the top p channels to retain. They apply the optimization process on each layer to obtain the masks β_i, use these to prune and then perform fine-tuning at the end.

An empirical evaluation of CCP is carried out by pruning the ResNet models trained on the CIFAR10 and ImageNet data, and the results compared to several methods including: pruning small filters, ThinNet, optimizing channel pruning, Soft Filter pruning [89], NISP [90] and AutoML [109]. For CIFAR10, the experiments are carried out with a pruning rate of 35% and 40%, and in each case, CCP has a smaller drop in accuracy (0.04% and 0.08% respectively) than the other methods, with the exception of the method for pruning small filters which results in a small improvement in accuracy (0.02%). However, the pruning small filters method has a much lower reduction in the FLOPS (27.6%) in comparison to CCP (52.6%). The results for ImageNet show, that for similar reductions in FLOPS, CCP has less of a drop in accuracy than the other methods.

It's worth noting, that like EigenDamage, CCP is able to obtain good results without the need for an iterative process that uses fine-tuning after pruning each layer.

VI. COMPARISON OF PUBLISHED RESULTS

As the above sections describe, previous studies of pruning report results on varying data sets, architectures and methods that have evolved with time, making comparison of results across the different studies difficult. The survey provides a resource in the form of a pivot table that can be used by the community to explore the reported performance of over

TABLE 2. Number of reported results for a given architecture and data set.

Architecture	CALTECH256	CamVid	CIFAR10	CIFAR100	CUB200-2011	Flowers102	ImageNet	MNIST	Pascal VOC	SVHN	**Total**
AlexNet							22				22
DenseNet-100			3								3
DenseNet-40			3	1						2	3
FCN32									1		1
LeNet-300								9			9
LeNet5								23			23
Res101							6				6
Res110			13								13
Res152							3				3
Res164			1	1							1
Res18	1		4		1	1	3				8
Res20			6								6
Res32			17	10							17
Res34							3				3
Res50							26				26
Res56			20				1				21
SegNet	1									1	
VGG16	1		13	1	3	1	10				24
VGG19			12	12	1						13
Total	1	1	48	13	3	1	59	28	1	2	115

50 methods on different architectures and data.[4] Table 2 shows how many times each combination of data and architecture has been used, indicating the wide variety of comparisons possible.

To illustrate the use of the resource, we use it to compare the reported results on two combinations of architecture and data for which there are a significant number of comparisons across different pruning methods, namely: (i) AlexNet and ResNet50 on ImageNet and (ii) ResNet56 and VGG16 on CIFAR10. Fig. 4 show the results reported in terms of the drop in Top-1 accuracy, and percent reduction in FLOPs or parameters where the labels used for the pruning methods are from the primary sources, with suffixes reflecting the variations in pruning methodology used. The main observations are that:

1) For AlexNet on ImageNet, AOFP-B2 achieves a 41% reduction in FLOPs with a 0.46% increase in accuracy and Dyn Surg [65], SSR-L [56] and NeST [41] achieve over 93% reduction in parameters without loss in accuracy. Other methods that compromise accuracy do not necessarily result in a greater reduction in parameters.
2) In comparison to AlexNet, it is harder to prune ResNet50 on ImageNet, although AOFP-C1 achieves 33% reduction in FLOPs without affecting accuracy. As accuracy is compromised, there are methods that show significant reductions in parameters. These include KSE and ThinNet with reductions in parameters by 78% and 95%, with a decline of 0.64%, and 0.84% in accuracy, respectively.
3) When pruning ResNet56 on the CIFAR10 data, the methods KSE and SFP-NFT show reductions in FLOPs of 60% and 28% without compromising accuracy. For VGG16 on CIFAR10, AOFP, PF_EC and NetSlimming result in a 75%, 63%, and 51% reduction in flops respectively without reductions in accuracy. For both networks, it appears to be difficult to gain further reductions (beyond KSE and AOFP) even when compromising accuracy.
4) The charts show that several methods are able to reduce FLOPs and parameters without compromising accuracy and aid generalizability (e.g., AOFP, SFP, NetSlimming), though compromising accuracy a little can sometimes lead to more significant reductions in FLOPs and parameters.
5) When looking at results within methods, it is possible to confirm our expectation that compromising accuracy can result in greater reductions in parameters and FLOPS (e.g, see results for Filter Sketch and FPGM for ResNet50 in Fig. 4). However, this trade-off is not evident when considering results across different pruning methods.

[4]This resource is available from https://1drv.ms/x/s!ArCIJ6nceQY3-BWxGMfJRbmhyUK_?e=WZUyhm

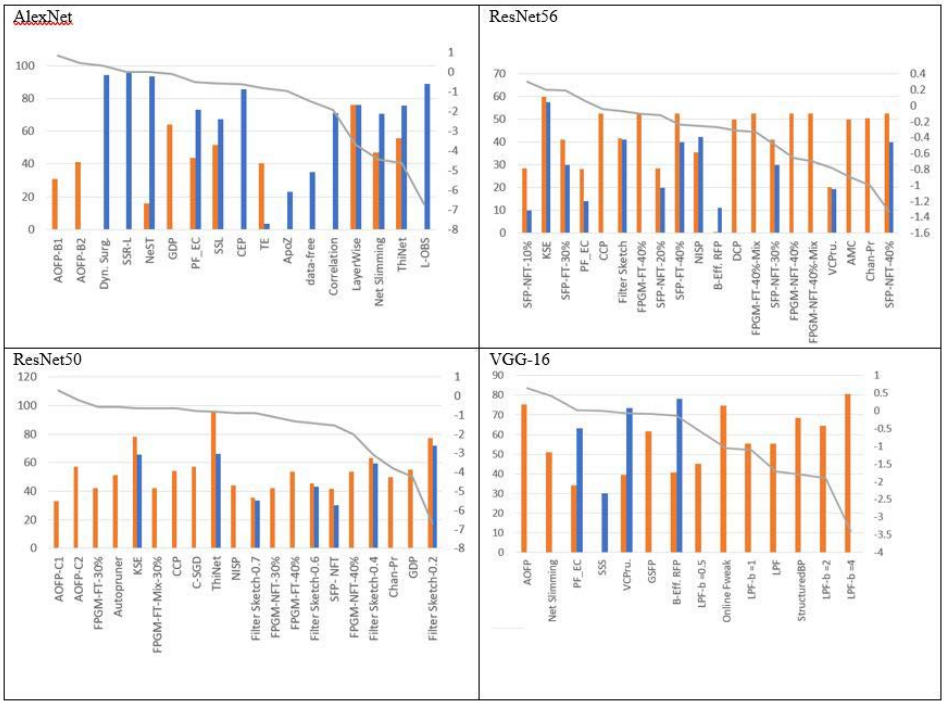

FIGURE 4. Results of pruning AlexNet and, ResNet50 on ImageNet (left column), and ResNet56 and VGG16 on the CIFAR10 data (right column). Charts show the percent reduction in parameters where available (blue bars, left axis) and FLOPs (orange bars, left axis), and reduction in baseline Top-1 accuracy (grey line, right axis). The labels used for the methods are from the primary sources, with suffixes reflecting the variations in pruning rates.

6) Although AOFP does perform well in retaining accuracy for three of the four cases, in general, the performance of the methods varies depending on the architecture and the data set.

VII. CONCLUSION AND FUTURE WORK

This paper has presented a survey of methods for pruning neural networks, focusing on methods based on magnitude pruning, use of similarity and clustering, and methods based on sensitivity analysis.

Magnitude based pruning methods have developed from removal of small weights in MLPs to methods for pruning filters and channels which lead to substantial reductions in the size of deep networks. The range of methods developed include: (i) those that are data dependent and use examples to assess the relevance of output channels, (ii) methods that are independent of data, which assess the contributions of filters and channels directly, and (iii) methods that utilise optimization algorithms to find filters that approximate channels.

Methods based on sensitivity analysis are the most transparent in that they are based on approximating the loss due to changes to a network. The development of methods based on a Taylor series approximation represents the primary line of research in this category of methods. Different studies have adopted different assumptions in order to make the computation of the Taylor approximation feasible. In one of the first studies, the OBD method assumed a diagonal Hessian matrix, ignoring both first order gradients and second order non-diagonal gradients. This was followed by OBS, a method that aimed to take account of non-diagonal elements of the Hessian but has been shown to struggle when pruning multiple weights that are correlated. The EigenDamage method aims to take better account of correlations by approximating the Hessian with a Fisher Information Matrix and using a reparameterization to a new space in which the weights are approximately independent. In an alternative approach, the Collaborative Channel Pruning (CCP) method formulates the pruning task as a quadratic programming problem. Molchanov *et al.* [105] develop a method based on a first-order approximation, arguing that the variance in the loss, as training approaches a local solution, is an indicator of stability and provides a good measure of the importance of filters. In contrast to most of the other methods that adopt layer by layer pruning with fine tuning after each layer, both EigenDamage and CCP show that it is possible to obtain good results with one-shot pruning followed by fine-tuning. These three recent methods all show good results on large scale networks and data sets, though direct empirical comparisons between them have yet to be published. The survey also found two alternatives to use of Taylor series approximations: a method that aims to learn which filters to prune [93] and a method based on the use of multi-armed bandits [118], both of which have the potential to explore new avenues of research on pruning methods.

The survey reveals a number of positive results about the Lottery Hypothesis: Lotteries appear to perform well in transfer learning, and lotteries exist for tasks such as NLP and for architectures such as LSTMs. Lotteries even seem to

be independent of the type of optimizer used during training. Much of the current research on lotteries is based on deep networks, but it is interesting that one of the earliest papers in the field demonstrates the need to overparameterize a small feedforward network for modelling a 4-bit multiplexor. Thus, it might prove fruitful to explore the properties of lotteries on smaller problems as well as the larger networks of today. The existence of good lotteries does appear to depend on the fine-tuning process adopted and an interesting observation, that challenges some of the empirical studies reported, is that even random pruning can achieve good results following fine-tuning [119], so further studies of how the remaining weights compensate for those that are removed, could results in new insights. Although studies on lotteries provide valuable insight, further research on specialist hardware and libraries is needed for methods that prune individual weights to become practical [120].

The survey found least research on methods that use similarity and clustering to develop pruning methods. A method that utilised a cosine similarity measure concluded that it was more suitable for MLPs than CNNs, while a method that utilises agglomerative clustering of filters results in up to a 3-fold reduction on ResNet when it is applied to ImageNet. These results suggest there is merit in developing a more theoretical understanding of the functional equivalence of different classes of deep networks, analogous to the studies on equivalence of MLPs [121].

Given the different approaches to pruning, some may be complimentary, and there is some evidence that combining them might result in further compression of networks. For example,He *et al.* [94] present results showing that combining their method based on the use of Lasso regression with factorization results in additional gains, and Han *et al.* [98] use a pipeline of magnitude pruning, clustering and Huffman coding to increase the level of compression that can be achieved.

One of the challenges in making sense of the empirical evaluations reported in the papers surveyed is that, as new deep learning architectures have developed and as new methods have been published, the comparisons carried out have evolved. The survey has therefore collated the published results of over 50 methods for a variety of data sets and architecture that is available as a resource for other researchers. Section VI uses this resource to present the first comprehensive comparison of published results across different pruning methods for different architectures. The comparison of published results shows that significant reductions can be obtained for AlexNet, ResNet and VGG, though there is no single method that is best, and that it is harder to prune ResNet than the other architectures. One can hypothesize that its use of skip connections makes it more optimal, though this is something that needs exploring. Likewise, given that different methods seem best for different architectures, it is worth studying and developing methods for specific architectures. The data also reveals that there are limited evaluations on other networks such as the InceptionNet, DenseNet, SegNet, FCN32 and datasets such as CIFAR100, Flowers102, CUB200-2011 (see Table 2). A comprehensive independent evaluation of the methods that includes consideration of the issues raised by the Lottery hypothesis across a wider range of data and architectures would be a useful advance in the field. In conclusion, this survey has presented the key research directions in pruning neural networks by summarizing how the field has progressed from the early algorithms that focused on small fully connected networks to the much larger deep neural networks of today. The survey has aimed to highlight the motivations and insights identified in the papers, and provides a resource for comparison of the reported results, architectures and data sets used in several studies which we hope will be useful to researchers in the field.

APPENDIX A
SUMMARY OF DATA SETS USED IN COMPARING PRUNING METHODS

MNIST [61]: The MNIST (Modified National Institute of Standards and Technology) data set consists of handwritten 28×28 images of digits. It has 60,000 examples of training data and 10,000 examples for the test set.

PASCAL VOC [198]: The PASCAL VOC data sets have formed the basis of an annual competitions from 2005 to 2012. The VOC 2007 data annotates objects in 20 classes and consists of 9,963 images and 24,640 annotated objects. The VOC 2012 data, which consists of 11530 images, are annotated with 27450 regions of interest and 6929 segmentations.

CamVid [199]: CamVid (Cambridge-driving Labelled Video Database) is a data set with videos captured from an automobile. In total over 10mins of video is provided together with over 700 images from the videos that have been labelled. Each pixel of an image is labelled to indicate if it is part of an objects in one of 32 semantic classes.

Oxford-Flowers [200]: The Oxford-Flowers data consists of 102 classes of common flowers in the UK. It provides 2040 training images and 6129 images for testing.

LFW [201]: The LFW (Labelled Faces in the Wild) is one of the largest and widely used data sets to evaluate face recognition algorithms. It includes 250×250 pixel images of over 5.7K individuals, with over 13K images in total.

CIFAR-10 &100 [202]: The CIFAR-10 (Canadian Institute for Advanced Research) data set is a collection of 32×32 colour images in 10 different classes. The data set splits into two sets: 50,000 images for training and 10,000 for testing. CIFAR-100 is similar to CIFAR-10 but has 100 classes, where each class has 500 training images and 100 test images.

ImageNet [203]: ImageNet contains millions of images organized using the WordNet hierarchy. It has over 14M images classified in over 21K groups and has provided the data sets for the ImageNet Large Scale Visual Recognition Challenges (ILSVRC) held since 2010.

It is one of the most widely used data sets in benchmarking deep learning models and methods for pruning. A smaller subset known as TinyImageNet is sometimes used and is also available (https://tiny-imagenet.herokuapp.com). It consists of 200 classes with 500 training, 50 validation and 50 testing images per class.

SVHN [204]: The SVHN (Street View House Number) data set is a collection of 600K, 32×32 images of house numbers in Google Street View images. The data set provides 73,257 images for training and 26,032 for testing.

UCSD-Birds [205]: The UCSD-Birds data set provides 11788 images of birds, labelled as one of 200 different types of species. The data is split into training and testing sets of 5994 and 5794 respectively.

Places365 [206]: Places365 is a data set with 8 million 200×200 pixel images of scenes labeled with one of 434 categories, such as bridge, kitchen, boxing ring, etc. It provides 50 images per class for validation and the test set consist of 500 images per class.

CASIA-WebFace [207]: CASIA-WebFace is a data set that was created for evaluating face recognition systems. It provides over 494K images of over 10K individuals.

WMT'14 En2De [208]: WMT'14 En2De is one of the benchmark language data sets provided for a task set for the Workshop on Statistical Machine Translation held in 2014. This data set consists of 4.5M English-German pairs of sentences.

FashionMNIST [209]: FashionMNIST is an alternative to the MNIST data set with 28×28 images of fashion products classified in 10 categories. Like MNIST, there are 60,000 images for training and 10,000 images for testing,

APPENDIX B
SUMMARY OF NOTATION

- In general, we use X to denote input channels, W to denote weights of filters and Y to denote output channels.
- Y_i j is used to denote the output feature map obtained by applying a filter $W_{j,i}$ on input channels X_i
- w_i, w_j, $w_{j,i}$ are used to denote individual weights.
- β is used to denote a binary mask where $\beta_i = 1$ indicates that a feature map or filter should be retained and $\beta = 0$ indicates that it should be removed.
- $\mathcal{L}$ is used to denotes a loss function
- L_0, L_1, L_2 denote norms, with L_0 counting non-zero values, L_1, being the sum of absolute. values, L_2 being the square root of the sum of squares (Euclidean distance).
- $\|W\|_n$ will be used to indicate the use of a norm in an equation with the an equation, with the subscript n indicating the specific norm.
- $\|W\|_F$, known as the Frobenius norm is sometimes used to denote the application of the Euclidean distance to the elements of a matrix

REFERENCES

[1] S. Kuutti, R. Bowden, Y. C. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2020.
[2] S. M. McKinney, M. Sieniek, V. Godbole, N. Antropova, H. Ashrafian, T. Back, M. Chesus, C. GC, A. Darzi, M. Etemadi, F. Garcia-Vicente, F. Gilbert, M. Halling-Brown, D. Hassabis, and S. Jansen, "International evaluation of an AI system for breast cancer screening," *Nature*, vol. 577, no. 7788, pp. 89–94, Jan. 2020.
[3] G. Hinton, D. Y. Deng, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Process. Mag.*, vol. 29, pp. 82–97, 2012.
[4] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, Feb. 2021.
[5] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, p. 36, 2019.
[6] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 48, pp. 30033–30038, Dec. 2020.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[9] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 1988, pp. 107–115.
[10] J. K. Kruschke, "Creating local and distributed bottlenecks in hidden layers of back-propagation networks," in *Proc. Connectionist Models Summer School*, 1988, pp. 120–126.
[11] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 740–747, 1993.
[12] Y. Chauvin, "A back-propagation algorithm with optimal use of hidden units," in *Proc. Adv. Neural Inf. Process. Syst.*, 1988, pp. 519–526.
[13] D. E. Weigend, "Back-propagation, weight-elimination and time series prediction," in *Proc. Connectionist Models Summer School*, 1990, pp. 105–116.
[14] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination applied to currency exchange rate prediction," in *Proc. Seattle Int. Joint Conf. Neural Netw. (IJCNN)*, Nov. 1991, pp. 837–841.
[15] Z. Zhou, W. Zhou, R. Hong, and H. Li, "Online filter weakening and pruning for efficient convnets," in *Proc. IEEE Int. Conf. Multimedia Expo. (ICME)*, Jul. 2018, pp. 1–6.
[16] A. M. Chen, H.-M. Lu, and R. Hecht-Nielsen, "On the geometry of feedforward neural network error surfaces," *Neural Comput.*, vol. 5, no. 6, pp. 910–927, Nov. 1993.
[17] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," 2016, *arXiv:1602.01528*.
[18] L. Li, J. Zhu, and M.-T. Sun, "Deep learning based method for pruning deep neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo. Workshops (ICMEW)*, Jul. 2019, pp. 312–317.
[19] A. RoyChowdhury, P. Sharma, E. Learned-Miller, and A. Roy, "Reducing duplicate filters in deep neural networks," in *Proc. NIPS Workshop Deep Learning: Bridging Theory Pract.*, 2017, pp. 1–7.
[20] H. J. Sussmann, "Uniqueness of the weights for minimal feedforward nets with a given input-output map," *Neural Netw.*, vol. 5, no. 4, pp. 589–593, Jul. 1992.
[21] Z. Zhou, W. Zhou, H. Li, and R. Hong, "Online filter clustering and pruning for efficient convnets," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 11–15.
[22] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage," in *Proc. Neural Inf. Process. Syst.*vol. 89, D. S. Touretzky, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1990, pp. 506–598.
[23] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Proc. IEEE Int. Conf. Neural Netw.*, Mar. 1993, pp. 293–299.
[24] B. Hassibi, D. G. Stork, G. Wolff, and T. Watanabe, "Optimal brain surgeon: Extensions and performance comparison," in *Proc. Neural Inf. Process. Syst.*, 1993, pp. 263–279.

[25] J. P. Cohen, H. Z. Lo, and W. Ding, "RandomOut: Using a convolutional gradient norm to rescue convolutional filters," 2016, *arXiv:1602.05931*.
[26] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," 2018, *arXiv:1810.02340*.
[27] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, "Accelerating convolutional networks via global & dynamic filter pruning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2425–2432.
[28] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, pp. 535–541. ACM, 2006.
[29] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[30] G. Urban, J. K. Geras, S. E. Kahou, S. Aslan, R. C. Wang, A. Mohamed, M. Philipose, and M. Richardson, "Do deep convolutional nets really need to be deep and convolutional," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.
[31] L. Zhang, Z. Tan, J. Song, J. Chen, C. Bao, and K. Ma, "SCAN: A scalable neural networks framework towards compact and efficient models," 2019, *arXiv:1906.03951*.
[32] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6655–6659.
[33] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," 2014, *arXiv:1405.3866*.
[34] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4350–4359.
[35] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," 2017, *arXiv:1702.03044*.
[36] Y. Zhao, X. Gao, D. Bates, R. Mullins, and C.-Z. Xu, "Focused quantization for sparse CNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5585–5594.
[37] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*.
[38] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.
[39] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2704–2713.
[40] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, Nov. 2017, *arXiv:1611.02167*.
[41] X. Dai, H. Yin, and N. K. Jha, "NeST: A neural network synthesis tool based on a grow-and-prune paradigm," *IEEE Trans. Comput.*, vol. 68, no. 10, pp. 1487–1497, Oct. 2019.
[42] X. Li, Y. Zhou, Z. Pan, and J. Feng, "Partial order pruning: For best speed/accuracy trade-off in neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9145–9153.
[43] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, "MetaPruning: Meta learning for automatic neural network channel pruning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3295–3304.
[44] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, "Channel pruning via automatic structure search," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 673–679.
[45] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2423–2432.
[46] B. Zoph and V. Q. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2017.
[47] J. Chung and T. Shin, "Simplifying deep neural networks for neuromorphic architectures," in *Proc. 53rd Annu. Design Autom. Conf.*, Jun. 2016, pp. 1–6.
[48] K. Goetschalckx, B. Moons, P. Wambacq, and M. Verhelst, "Efficiently combining SVD, pruning, clustering and retraining for enhanced neural network compression," in *Proc. 2nd Int. Workshop Embedded Mobile Deep Learn.*, Jun. 2018, pp. 1–6.
[49] P. K. Gadosey, Y. Li, and P. T. Yamak, "On pruned, quantized and compact CNN architectures for vision applications: An empirical study," in *Proc. Int. Conf. Artif. Intell., Inf. Process. Cloud Comput. (AIIPCC)*, 2019, pp. 1–8.
[50] V. Lebedev and V. Lempitsky, "Speeding-up convolutional neural networks: A survey," *Bull. Polish Acad. Sci. Tech. Sci.*, vol. 66, no. 6, pp. 799–810, 2018.
[51] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.
[52] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," 2021, *arXiv:2106.08962*.
[53] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 2003.
[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[55] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 304–320.
[56] S. Lin, "Toward compact convnets via structure-sparsity regularized filter pruning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 574–588, Feb. 2019.
[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.
[58] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. 30th Int. Conf. Neural Inf.*, 2016, pp. 2082–2090.
[59] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2780–2789.
[60] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.
[61] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[63] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, 2019.
[64] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," 2015, *arXiv:1506.02626*.
[65] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1387–1395.
[66] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019, *arXiv:1803.03635*.
[67] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proc. 7th Int. Conf. Learn. Represent., (ICLR)*, New Orleans, LA, USA, May 2019, *arXiv:1810.05270*.
[68] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Stabilizing the lottery ticket hypothesis," 2019, *arXiv:1903.01611*.
[69] S. Ari Morcos, H. Yu, M. Paganini, and Y. Tian, "One ticket to win them all: Generalizing lottery ticket initializations across datasets and optimizers," in *Proc. Neural Inf. Process. Syst. (NeuralPS)*, 2019, pp. 4932–4942.
[70] N. Hubens, M. Mancas, M. Decombas, M. Preda, T. Zaharia, B. Gosselin, and T. Dutoit, "An experimental study of the impact of pre-training on the pruning of a convolutional neural network," in *Proc. 3rd Int. Conf. Appl. Intell. Syst.*, Jan. 2020, pp. 1–6.
[71] H. Yu, S. Edunov, Y. Tian, and A. S. Morcos, "Playing the lottery with rewards and multiple languages: Lottery tickets in RL and NLP," 2019, *arXiv:1906.02768*.
[72] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *Proc. Int. Conf. Learn. Represent.*, 2017, *arXiv:1609.07843*.
[73] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. Aidan Gomez, U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA: Curran Associates, 2017, pp. 6000–6010.
[74] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*.

[75] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2015.
[76] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. 5th Int. Conf. Learn. Represent., (ICLR)*, Toulon, France, Apr. 2017, *arXiv:1608.08710*.
[77] M. Denil, B. Shakibi, L. Dinh, and N. D. Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.
[78] B. O. Ayinde, T. Inanc, and J. M. Zurada, "Redundant feature pruning for accelerated inference in deep neural networks," *Neural Netw.*, vol. 118, pp. 148–158, Oct. 2019.
[79] A. Polyak and L. Wolf, "Channel-level acceleration of deep face representations," *IEEE Access*, vol. 3, pp. 2163–2175, 2015.
[80] M. Lin, Q. Chen, and S. Yan, "Network in a network," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, *arXiv:1312.4400*.
[81] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, Oct. 2015.
[82] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*.
[83] J.-H. Luo and J. Wu, "An entropy-based pruning method for CNN compression," 2017, *arXiv:1706.05791*.
[84] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*.
[85] M. Lin, L. Cao, S. Li, Q. Ye, Y. Tian, J. Liu, Q. Tian, and R. Ji, "Filter sketch for network pruning," 2020, *arXiv:2001.08514*.
[86] E. Liberty, "Simple and deterministic matrix sketching," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 581–588.
[87] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4340–4349.
[88] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.
[89] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2234–2240.
[90] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.
[91] X. Ding, G. Ding, Y. Guo, J. Han, and C. Yan, "Approximated Oracle filter pruning for destructive CNN width optimization," 2019, *arXiv:1905.04748*.
[92] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2755–2763.
[93] Q. Huang, K. Zhou, S. You, and U. Neumann, "Learning to prune filters in convolutional neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 709–718.
[94] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1398–1406.
[95] H. Wang, Q. Zhang, Y. Wang, and H. Hu, "Structured probabilistic pruning for convolutional neural network acceleration," 2017, *arXiv:1709.06994*.
[96] J.-H. Luo and J. Wu, "AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference," 2018, *arXiv:1805.08941*.
[97] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," 2018, *arXiv:1802.00124*.
[98] S. Han, H. Mao, and W. J. Dally, "A deep neural network compression pipeline: Pruning, quantization, Huffman encoding," in *Proc. ICLR*, 2016, *arXiv:1510.00149v5*.
[99] S. Son, S. Nah, and K. Lee, "Clustering convolutional kernels to compress deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 225–240.
[100] Y. Li, S. Lin, B. Zhang, J. Liu, D. Doermann, Y. Wu, F. Huang, and R. Ji, "Exploiting kernel sparsity and entropy for interpretable CNN compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2800–2809.
[101] C. Wang, R. Grosse, S. Fidler, and G. Zhang, "EigenDamage: Structured pruning in the kronecker-factored eigenbasis," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6566–6575.
[102] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. New York, NY, USA: Springer, 2006.
[103] S. Thrun, "The MONK's problems—A performance comparison of different learning algorithms," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-91-197, 1991.
[104] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce English text," *Complex Syst.*, vol. 1, pp. 145–168, Feb. 1987.
[105] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, *arXiv:1611.06440*.
[106] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11264–11272.
[107] R. Grosse and J. Martens, "A Kronecker-factored approximate Fisher matrix for convolution layers," in *Proc. Int. Conf. Mach. Learn.*, Feb. 2016, pp. 573–582.
[108] H. Peng, J. Wu, S. Chen, and J. Huang, "Collaborative channel pruning for deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5113–5122.
[109] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: Automl for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 784–800.
[110] C. Jiang, G. Li, C. Qian, and K. Tang, "Efficient DNN neuron pruning by minimizing layer-wise nonlinear reconstruction error," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2298–2304.
[111] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA: Curran Associates, 2017, pp. 4860–4874.
[112] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 875–886.
[113] K. Xu, X. Wang, Q. Jia, J. An, and D. Wang, "Globally soft filter pruning for efficient convolutional neural networks," Tech. Rep., 2018. [Online]. Available: https://paperswithcode.com/paper/globally-soft-filter-pruning-for-efficient
[114] K. Neklyudov, D. Molchanov, A. Ashukha, and D. Vetrov, "Structured Bayesian pruning via log-normal multiplicative noise," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA: Curran Associates, 2017, pp. 6778–6787.
[115] R. Bao, X. Yuan, Z. Chen, and R. Ma, "Cross-entropy pruning for compressing convolutional neural networks," *Neural Comput.*, vol. 30, no. 11, pp. 3128–3149, Nov. 2018.
[116] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," 2015, *arXiv:1507.06149*.
[117] Y. Sun, X. Wang, and X. Tang, "Sparsifying neural network connections for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4856–4864.
[118] S. Ameen and S. Vadera, "Pruning neural networks using multi-armed bandits," *Comput. J.*, vol. 63, no. 7, pp. 1099–1108, Jul. 2020.
[119] D. Mittal, S. Bhardwaj, M. M. Khapra, and B. Ravindran, "Studying the plasticity in deep convolutional neural networks using random pruning," *Mach. Vis. Appl.*, vol. 30, no. 2, pp. 203–216, Mar. 2019.
[120] E. Elsen, M. Dukhan, T. Gale, and K. Simonyan, "Fast sparse ConvNets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14629–14638.
[121] V. Kůrková and P. C. Kainen, "Functionally equivalent feedforward neural networks," *Neural Comput.*, vol. 6, no. 3, pp. 543–558, May 1994.
[122] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 177–185.
[123] A. Graves, "Practical variational inference for neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2348–2356.
[124] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-trim: Convex pruning of deep neural networks with performance guarantee," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA: Curran Associates, 2017, pp. 3180–3189.
[125] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017, *arXiv:1710.01878*.

[126] C. Chen, F. Tung, N. Vedula, and G. Mori, "Constraint-aware deep neural network compression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 400–415.
[127] D. Lee, S. Kang, and K. Choi, "ComPEND: Computation pruning through early negative detection for ReLU in a deep neural network accelerator," in *Proc. Int. Conf. Supercomput.*, Jun. 2018, pp. 139–148.
[128] G. Li, C. Qian, C. Jiang, X. Lu, and K. Tang, "Optimization based layer-wise magnitude-based pruning for DNN compression," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2383–2389.
[129] C. Liu and Q. Liu, "Improvement of pruning method for convolution neural network compression," in *Proc. 2nd Int. Conf. Deep Learn. Technol. (ICDLT)*, 2018, pp. 57–60.
[130] Z. Qin, F. Yu, C. Liu, and X. Chen, "Demystifying neural network filter pruning," 2018, *arXiv:1811.02639*.
[131] R. Yazdani, M. Riera, J.-M. Arnau, and A. Gonzalez, "The dark side of DNN pruning," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 790–801.
[132] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic DNN weight pruning framework using alternating direction method of multipliers," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 184–199.
[133] X. Ding, G. DIng, X. Zhou, Y. Guo, J. Han, and J. Liu, "Global sparse momentum SGD for pruning very deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6379–6391.
[134] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," 2019, *arXiv:1907.04840*.
[135] S. Gui, H. N. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1283–1294.
[136] K. Helwegen, J. Widdicombe, L. Geiger, Z. Liu, K.-T. Cheng, and R. Nusselder, "Latent weights do not exist: Rethinking binarized neural network optimization," 2019, *arXiv:1906.02107*.
[137] L. Hou, J. Zhu, J. Kwok, F. Gao, T. Qin, and T.-Y. Liu, "Normalization helps training of quantized LSTM," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7344–7354.
[138] K. Lee, H. Kim, H. Lee, and D. Shin, "Flexible group-level pruning of deep neural networks for fast inference on mobile CPUs: Work-in-progress," in *Proc. Int. Conf. Compliers, Archit. Synth. Embedded Syst. Companion (CASES)*, 2019, pp. 1–2.
[139] J. Li, Q. Qi, J. Wang, C. Ge, Y. Li, Z. Yue, and H. Sun, "OICSR: Out-in-channel sparsity regularization for compact deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7046–7055.
[140] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3D deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 963–973.
[141] J. Song, Y. Chen, X. Wang, C. Shen, and M. Song, "Deep model transferability from attribution maps," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6179–6189.
[142] Y. Xu, Y. Wang, J. Zeng, K. Han, X. U. Chunjing, D. Tao, and C. Xu, "Positive-unlabeled compression on the cloud," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2561–2570.
[143] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing lottery tickets: Zeros, signs, and the supermask," 2019, *arXiv:1905.01067*.
[144] Y. Zhu, C. Li, B. Luo, J. Tang, and X. Wang, "Dense feature aggregation and pruning for RGBT tracking," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 465–472.
[145] T. Kim, D. Ahn, and J.-J. Kim, "V-LSTM: An efficient LSTM accelerator using fixed nonzero-ratio viterbi-based pruning," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2020, p. 326.
[146] Q. Li, C. Li, and H. Chen, "Incremental filter pruning via random walk for accelerating deep convolutional neural networks," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 358–366.
[147] W. Niu, X. Ma, S. Lin, S. Wang, X. Qian, X. Lin, Y. Wang, and B. Ren, "PatDNN: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 907–922.
[148] A. Dubey, M. Chatterjee, and N. Ahuja, "Coreset-based neural network compression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 454–470.
[149] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal SGD for pruning very deep convolutional networks with complicated structure," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4943–4953.
[150] M. Lin, R. Ji, B. Chen, F. Chao, J. Liu, W. Zeng, Y. Tian, and Q. Tian, "Training compact CNNs for image classification using dynamic-coded filter fusion," 2021, *arXiv:2107.06916*.
[151] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, San Mateo, CA, USA: Morgan Kaufmann, 1992, pp. 164–171.
[152] J. Xu and W. C. Daniel Ho, "A node pruning algorithm based on optimal brain surgeon for feedforward neural networks," in *Proc. 3rd Int. Conf. Adv. Neural Netw. (ISNN)*, vol. 1. Berlin, Germany: Springer, 2006, pp. 524–529.
[153] C. Endisch, C. Hackl, and D. Schröder, "Optimal brain surgeon for general dynamic neural networks," in *Proc. Aritficial Intell. 13th Portuguese Conf. Prog. Artif. Intell. (EPIA)*, Berlin, Germany: Springer, 2007, pp. 15–28.
[154] C. Endisch, P. Stolze, P. Endisch, C. Hackl, and R. Kennel, "Levenberg–Marquardt-based OBS algorithm using adaptive pruning interval for system identification with dynamic neural networks," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2009, pp. 3402–3408.
[155] S. Ameen, "Optimizing deep learning networks using multi-armed bandits," Ph.D. thesis, School Comput., Sci. Eng., Univ. Salford, Greater Manchester, U.K., 2017.
[156] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, pp. 1–18, Feb. 2017.
[157] J. Guo and M. Potkonjak, "Pruning filters and classes: Towards on-device customization of convolutional neural networks," in *Proc. 1st Int. Workshop Deep Learn. Mobile Syst. Appl. (EMDL)*, 2017, pp. 13–17.
[158] M. A. Carreira-Perpinan and Y. Idelbayev, "'Learning-compression' algorithms for neural net pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8532–8541.
[159] L. N. Huynh, Y. Lee, and R. K. Balan, "D-pruner: Filter-based pruning method for deep convolutional neural network," in *Proc. 2nd Int. Workshop Embedded Mobile Deep Learn.*, Jun. 2018, pp. 7–12.
[160] S. Chen, L. Lin, Z. Zhang, and M. Gen, "Evolutionary NetArchitecture search for deep neural networks pruning," in *Proc. 2nd Int. Conf. Algorithms, Comput. Artif. Intell.*, Dec. 2019, pp. 189–196.
[161] W. Deng, X. Zhang, F. Liang, and G. Lin, "An adaptive empirical Bayesian method for sparse deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5564–5574.
[162] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "DeepSZ: A novel framework to compress deep neural networks by using error-bounded lossy compression," in *Proc. 28th Int. Symp. High-Perform. Parallel Distrib. Comput.*, Jun. 2019, pp. 159–170.
[163] H. Li, N. Liu, X. Ma, S. Lin, S. Ye, T. Zhang, X. Lin, W. Xu, and Y. Wang, "ADMM-based weight pruning for real-time deep learning acceleration on mobile devices," in *Proc. Great Lakes Symp. VLSI*, May 2019, pp. 501–506.
[164] Z. Qin, F. Yu, C. Liu, and X. Chen, "CAPTOR: A class adaptive filter pruning framework for convolutional neural networks in mobile applications," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, Jan. 2019, pp. 444–449.
[165] X. Xiao, Z. Wang, and S. Rajasekaran, "AutoPrune: Automatic network pruning by regularizing auxiliary parameters," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13681–13691.
[166] Z. Bao, J. Liu, and W. Zhang, "Using distillation to improve network performance after pruning and quantization," in *Proc. 2nd Int. Conf. Mach. Learn. Mach. Intell.*, Sep. 2019, pp. 3–6.
[167] C. Lemaire, A. Achkar, and P.-M. Jodoin, "Structured pruning of neural networks with budget-aware regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9108–9116.
[168] X. Dong and Y. Yang, "Network pruning via transformable architecture search," 2019, *arXiv:1905.09717*.
[169] S. Kundu and S. Sundaresan, "AttentionLite: Towards efficient self-attention models for vision," 2020, *arXiv:2101.05216*.
[170] P. Kaliamoorthi, A. Siddhant, E. Li, and M. Johnson, "Distilling large language models into tiny and effective students using pQRNN," 2021, *arXiv:2101.08890*.
[171] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," 2014, *arXiv:1412.6553*.
[172] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, "Holistic CNN compression via low-rank decomposition with knowledge transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 12, pp. 2889–2905, Dec. 2019.

[173] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1526–1535.
[174] V. Vanhoucke, A. Senior, and Z. Mark Mao, "Improving the speed of neural networks on cpus," in *Proc. Deep Learn. Unsupervised Feature Learn. Workshop, Neural Inf. Process. Syst.*, 2011. [Online]. Available: https://research.google/pubs/pub37631/
[175] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.
[176] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014, *arXiv:1412.6115*.
[177] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, pp. 4114–4122, 2016.
[178] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," 2016, *arXiv:1603.05279*.
[179] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018, *arXiv:1806.08342*.
[180] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-domain dynamic pruning for convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1043–1053.
[181] F. Tung and G. Mori, "CLIP-Q: Deep network compression learning by in-parallel pruning-quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7873–7882.
[182] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7948–7956.
[183] S. Chen, W. Wang, and S. J. Pan, "MetaQuant: Learning to quantize by learning to penetrate non-differentiable quantization," in *Adv. Neural Inf. Process. Syst.*, 2019, pp. 3918–3928.
[184] P. Wang, Q. Chen, X. He, and J. Cheng, "Towards accurate post-training network quantization via bit-split and stitching," in *Proc. Mach. Learn. Res.*, vol. 119, 2020, pp. 9847–9856.
[185] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, "Training with quantization noise for extreme model compression," in *Proc. Int. Conf. Learn. Represent.*, 2021, *arXiv:2004.07320*.
[186] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[187] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA: Curran Associates, 2017, pp. 2178–2188.
[188] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, and E. Choi, "MorphNet: Fast & simple resource-constrained structure learning of deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1586–1595.
[189] C.-H. Hsu, S.-H. Chang, J.-H. Liang, H.-P. Chou, C.-H. Liu, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, and D.-C. Juan, "MONAS: Multi-objective neural architecture search using reinforcement learning," 2018, *arXiv:1806.10332*.
[190] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 19–34.
[191] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn.*, J. D. A. Krause, Ed., Jul. 2018, pp. 4095–4104.
[192] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artrificial Intgelligence*, vol. 33, 2019, pp. 4780–4798.
[193] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2815–2823.
[194] J. Zhang, X. Chen, M. Song, and T. Li, "Eager pruning: Algorithm and architecture support for fast training of deep neural networks," in *Proc. 46th Int. Symp. Comput. Archit.*, Jun. 2019, pp. 292–303.
[195] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 2943–2952.
[196] J. Cheng, P.-S. Wang, G. Li, Q.-H. Hu, and H.-Q. Lu, "Recent advances in efficient computation of deep convolutional neural networks," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 1, pp. 64–77, Jan. 2018.
[197] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.
[198] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
[199] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan. 2009.
[200] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2008, pp. 722–729.
[201] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Proc. Workshop Faces Real-Life'Images, Detection, Alignment, Recognit.*, 2008. [Online]. Available: http://vis-www.cs.umass.edu/lfw/lfw.pdf
[202] A. Krizhevsky, V. Nair, and G. Hinton. (2009). *The CIFAR-10 and CIFAR-100 Datasets*. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html
[203] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[204] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011. [Online]. Available: https://research.google/pubs/pub37648/
[205] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, *The CALTECH-UCSD Birds-200-2011 Dataset*, document CNS-TR-201, 2011.
[206] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 487–495.
[207] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," 2014, *arXiv:1411.7923*.
[208] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, "Findings of the 2014 workshop on statistical machine translation," in *Proc. 9th Workshop Stat. Mach. Transl.*, 2014, pp. 12–58.
[209] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

SUNIL VADERA received the Ph.D. degree in computer science from The University of Manchester, in 1992. He is currently a Professor of computer science at the University of Salford, U.K., where he has served in many leadership roles, including as the Dean and the Head of the School of Computing, Science and Engineering, from 2011 to 2019. He currently leads the AI Foundry Project at the University of Salford which supports SMEs to develop innovative products and services using AI. His main research interests include pruning deep networks and applications of AI. He is a fellow of the British Computer Society, a Chartered Engineer (C.Eng.), and a Chartered IT Professional (CITP). In 2014, he was awarded the U.K. BDO Best Indian Scientist and Engineer in recognition of his contributions to computing, science, and engineering.

SALEM AMEEN received the B.Eng. degree from the Department of Electrical and Electronic Engineering, University of Seventh April, Libya, in 1999, the M.Tech. degree in computer science and engineering from the Jaypee Institute of Information Technology, India, in 2009, and the Ph.D. degree in computer science from the University of Salford, in 2018. His main research interests include machine learning, deep learning, multi-armed bandits, image mining, and time series forecasting.