Pruning Deep Neural Networks for Green Energy-Efficient Models: A Survey

Jihene Tmamna[1], Emna Ben Ayed[1,2], Rahma Fourati[1,3*], Mandar Gogate[4], Tughrul Arslan[5], Amir Hussain[4] and Mounir Ben Ayed[1,6]

[1]Research Groups in Intelligent Machines, National Engineering School of Sfax (ENIS), University of Sfax, Sfax, BP 1173, 3038, Sfax, Tunisia.
[2]Industry 4.0 Research Lab, Polytech-Sfax (IPSAS), Avenue 5 August, Rue Said Aboubaker, 3002, Sfax, Tunisia.
[3] Faculty of Law, Economics and Management Sciences of Jendouba (FSJEGJ), Jendouba, Tunisia.
[4]School of Computing, Merchiston Campus, Edinburgh Napier University, Edinburgh, EH10 5DT, Scotland, UK.
[5] School of Engineering, The University of Edinburgh, Edinburgh, EH9 3FF, UK.
[6]Computer Sciences and Communication Department, Faculty of Sciences of Sfax, University of Sfax, Sfax, Tunisia.

*Corresponding author(s). E-mail(s): rahma.fourati@ieee.org;
Contributing authors: jihen.tmamna@enis.tn; emna.benayed.b@ieee.org; m.gogate@napier.ac.uk; T.Arslan@ed.ac.uk; a.hussain@napier.ac.uk; mounir.benayed@ieee.org;

Abstract

Background: Over the past few years, larger and deeper neural network models, in particular convolutional neural networks have emerged and constantly pushed state-of-the-art performance in various fields. However, the computations required by these models have increased at an exponential rate. Massive computations not only have negative effects on research inclusiveness and deployment on limited resource devices but also a surprisingly large carbon footprint. Green deep learning is an emerging research field that encourages researchers to focus on energy consumption and carbon emissions during model training and inference. The aim is to achieve new results with light and energy-efficient deep neural networks. Many techniques can be used to achieve this goal.
Methods: Relevant studies have found that deep models have redundant and useless parameters that affect the final result of the model, thus providing theoretical support for the pruning of these models. Therefore, the focus of this timely review paper is to first, systematically outline recent developments in convolutional network pruning methods. In particular, we provide preliminary background knowledge for researchers to help understand this interdisciplinary field.
Results: The challenges of available model pruning methods are also highlighted to provide a basis for future research directions.
Conclusion: This survey underscores the pressing need for the development of innovative metrics to weigh diverse pruning objectives effectively. Furthermore, it identifies the exploration of pruning techniques tailored to advanced deep learning models like CNN-LSTM and ConvLSTM as a burgeoning area ripe for further investigation in the realm of green learning research.

Keywords: Deep convolutional neural networks, Green deep learning, Network compression, Network pruning

1 Introduction

After decades of development, deep convolutional neural networks (CNNs) have achieved great success in a variety of artificial intelligence applications, including image classification [37, 131, 174], object detection [10, 33, 118], and natural language processing (NLP) [25, 54]. Generally, this great success usually comes at the cost of their more complex structures with up to billions of parameters, which are accompanied by high memory and computational requirements for both the training and inference phases.

Several classification networks have been proposed, such as AlexNet [63], VGG16 [131], GoogLeNet [136], MobileNets, and the object detection networks including YOLOv3-v4 [10, 33], SSD [90], Faster-RCNN [121], and image segmentation including U-Net. As noted by Allen *et al.* [6], such models tend to be over-parameterized and come with a substantial memory footprint. For instance, the VGG16 has around 138 million parameters and needs over 500 MB of storage space in addition to 15.5G Multiply-ACcumulate (MAC) to process a 224×224 pixels input image. Similarly, the YOLOv4 network has 162 layers, 256 MB of memory, and 64 million parameters. It needs 29 Giga FLOPs (Floating Point Operations Per Second) when processing an image of size 416×416. The U-Net model comprises 31 million parameters, requires 372.5 MB of storage space, and incurs 54.65 billion FLOPs (Floating Point Operations) when segmenting an image with a resolution of 512x512.

Major challenges arise from the increasingly large size of CNN models. One of the primary challenges arises from the substantial rise in energy consumption, resulting in a negative impact on the environment. Additionally, there is a growing need for environmentally friendly models that operate efficiently during both development and deployment. Finally, the size of CNN models can hinder on-device training and inference on limited-resource devices, like edge devices with limited battery life and computational resources [88]. As CNN applications increasingly shift towards mobile and embedded devices, the optimization of CNN architectures has gained significant popularity. There has been a notable focus on developing network architectures that are more efficient and require less memory.

To mitigate this problem, one approach is to compress deep neural networks. There has been substantial research in network compression, to reduce the resource requirements of CNNs and facilitate CNN deployment on resource-constrained devices. Generally, the vast domain of CNN compression methods can be roughly arranged into four main categories as shown in Fig. 1, namely low-rank approximation [7, 27, 57, 59, 68], knowledge distillation [3, 161–163, 173], network pruning [18, 28, 50, 66, 81, 91, 92, 104, 129], and quantization [9, 14, 24, 69, 108]. First, the low-rank approximation method focuses on factorizing the weights matrix into low-rank ones. Second, knowledge distillation uses a teacher-student strategy to extract information from trained deep networks and transfer it to smaller networks with less storage and computing. The third category is network pruning which aims to eliminate irrelevant parameters to reduce model resource requirements while overcoming the problem of overfitting. Finally, quantization reduces the number of bits in each parameter by converting it from a 32-bit floating point to a lower bit depth to reduce storage space.

Pruning stands out as one of the most popular methods for compressing CNNs. Its prominence in CNN research is evident from extensive experimental validation, showcasing its effectiveness in accelerating and compressing CNN architectures. Consequently, the proliferation of related works has made it challenging for newcomers to navigate the diverse landscape of pruning methodologies and embark on research effectively. This study endeavors to delineate prevailing trends and furnish guidance to practitioners seeking the most suitable network pruning approach. Neural network pruning holds pivotal significance in deep learning research, facilitating the reduction of model sizes and enhancing their efficacy for real-world applications. This survey seeks to furnish insights into various pruning methods and their impact across different network architectures. Through comprehensive analysis and trend identification, this work aims to assist practitioners in selecting the most appropriate pruning method tailored to their networks, thereby fostering improved outcomes.

To our knowledge, there are only a few survey studies on network pruning. After carefully reviewing those surveys, we discovered that some

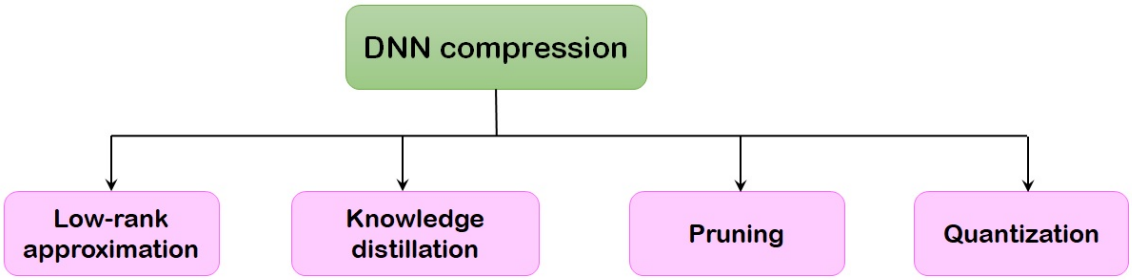

Fig. 1 Taxonomy of compression techniques

of the previous pruning techniques were not discussed. An overview of neural network pruning is provided by [87]who developed a comprehensive model that included pruning methods and evaluated the benefits and drawbacks of pruning procedures by covering only 26 research articles. The paper by Vadera and Ameen [143] gives an overview only of the methods used to select the irrelevant parameters. The surveys by [147, 151] only cover methods that reduce the number of weights in Deep Learning (DL) models during the initialization phase. Cong and Zhou [26] reviewed the typical CNNs models and network optimization methods. Different from the previous surveys, we integrated more pruning approaches into our analysis, where we gathered and discussed over 100 publications.

To collect published articles, we defined the period from 2015 to 2024 as the timeframe and considered the following publishers: *Elsevier*, *Springer*, *IEEE*, and *ACM Digital Library*. Additionally, we used Google Scholar and conference proceedings to identify more papers, employing search keywords such as "neural network pruning" and "acceleration of deep neural networks." We've observed a significant increase in research interest and demand for CNN pruning in recent years, largely motivated by the computational challenges associated with CNNs and their deployment on edge devices.

This paper aims to offer a comprehensive review of recent advances and literature in the field of network pruning, covering a wide array of pruning methods. We will conduct a performance study and discuss the benefits and drawbacks of various pruning factors and parameter selection methods. Our contributions may be summarized as follows:

1. The key factors presented in the field of network pruning are highlighted. Then, they are decomposed into three categories: pruning time, rate, and level.
2. The different pruning methods used to select the irrelevant parameters to identify their advantages and drawbacks are reviewed and classified.
3. All you need to evaluate a pruning method is detailed including the different datasets, models, and performance metrics.
4. The challenges and future research directions are raised for successful network pruning.

This survey is recommended for researchers interested in delving deeper into convolution-based network pruning. It offers insights into the history of this field, its current advancements, and the outstanding problems it faces.

This research paper is structured into five sections, as follows: Section 2 establishes the link between green learning and neural architecture search. Section 3 recalls the CNN architecture. The pruning process is presented in Section 4. Then, the key factors for designing a new pruning method are explained in detail in Section 5. Existing methods are presented according to three categories in Section 6. All you need to evaluate a pruning method is provided in Section 7. Section 8 explores future directions.

2 Genesis of DNN compression

Deep neural networks (DNNs) have recently succeeded in large-scale applications. This success has usually been achieved by increasingly large architectures requiring massive computations. According to [126], this trend is called red AI, where DNN

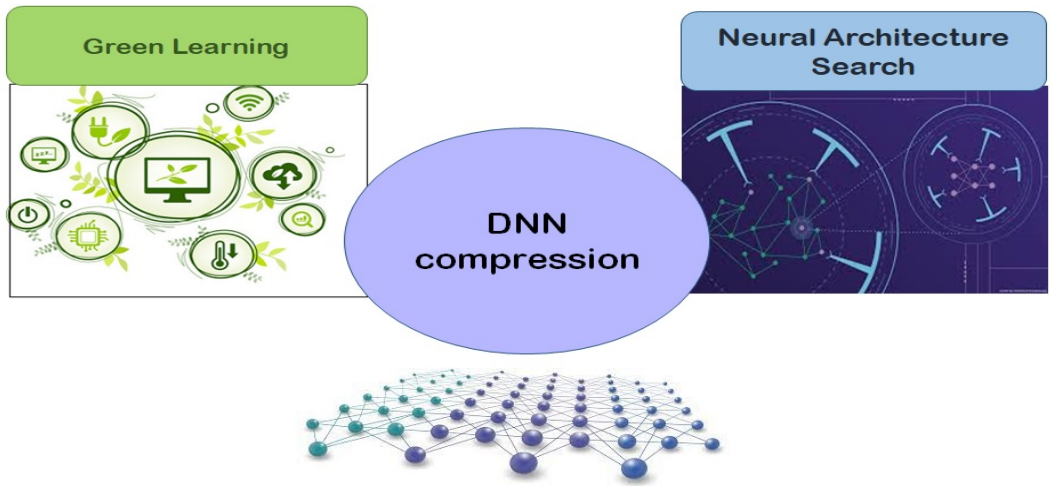

Fig. 2 DNN compression between GL and NAS

researchers focus on improving accuracy without considering cost or efficiency. For example, as reported in [126], the number of computations used to train DNNs models has increased 300,000x in 6 years. While improving model accuracy offers clear benefits, this trend not only leads to high costs but also contributes to an excessive carbon footprint. According to [134], training GPT-3 emits almost 500M carbons, equivalent to nearly five cars over their lifetime. To address these issues, Green deep learning, or Green AI, was first proposed by [126]. This trend aims to encourage AI researchers to obtain comparable or better results without increasing computational cost rather, ideally using as few computations as possible during model training and inference. In addition, some edge devices, such as Internet of Things devices and mobile devices with very few computational resources, require Green Deep Learning. Therefore, green DL is a necessary direction for future research.

In recent years, Neural Architecture Search (NAS), the field that automatically designs neural network architectures, has gained attention. NAS aims to achieve optimal performance with limited computing resources in an automated way, reducing human intervention. However, conventional NAS methods like reinforcement learning (RL) [178, 179] or evolutionary algorithms [86, 116, 117], as the search progresses, face the dilemma of demanding extensive computational resources. This strategy still leads to massive computations which not only have a surprisingly large carbon footprint but also have negative effects on research inclusiveness.

To achieve the goal of reducing energy usage and carbon emission during model design, training, and inference, several methods can be used. This paper focuses on presenting a systematic review of neural network pruning as a Green deep learning technology. Unlike previous conventional NAS approaches, the concept behind network pruning begins with an over-parameterized network that encompasses all candidate operations. The objective of neural network pruning is to look for lightweight and efficient models with few computational resources.

Fig. 2 depicts the positioning of DNN compression between Green Learning and NAS. The objectives of GL are a priority nowadays, especially with the climate variation around the world. While NAS methods aim to design an optimal neural network architecture for a specific problem, DNN compression leverages pre-trained models to achieve lightweight architectures.

3 CNNs background

Before delving into network pruning methods, let's first examine the CNN architecture to provide context for the rest of the paper.

The architecture of DNNs, which emulate the functioning of animal brains, relies on artificial neurons as their primary computational units. These neurons are organized into layers, with the

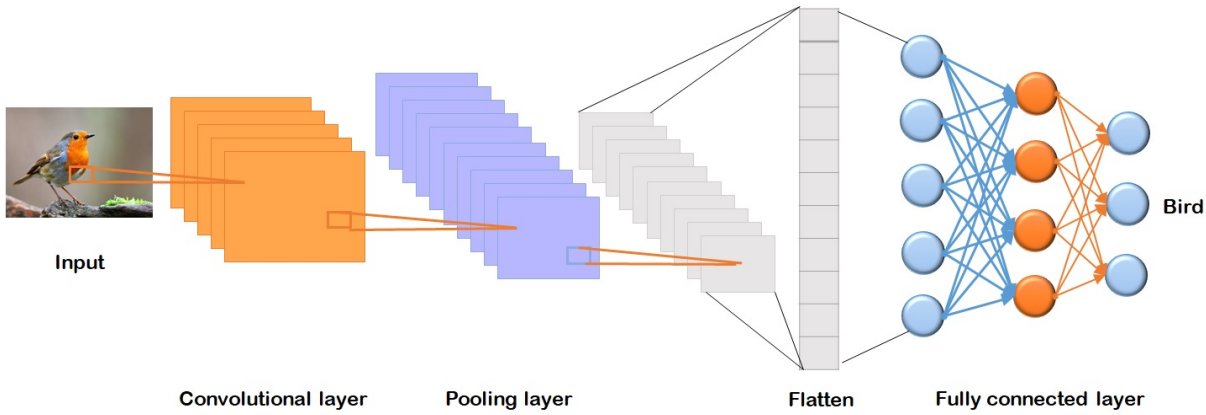

Fig. 3 CNN architecture

DNN architecture comprising multiple such layers. A deep convolutional neural network (CNN), a subtype of DNN, consists of three primary types of layers: convolution, pooling, and fully connected layers (as depicted in Fig.3). Convolution and pooling layers are considered hidden layers, while fully connected layers form the end layers. Convolutional layers, situated at the core of CNNs, consist of nodes known as filters, which perform convolution operations on the input data. Pooling layers process the feature maps generated by convolutional layers to reduce their dimensionality.

The architecture of CNNs has evolved over more than 30 years of research and development, primarily driven by the design of advanced modules and extensive recognition capabilities. These enhanced recognition capabilities contribute to improved network efficiency in derivative applications like image and video recognition. Nevertheless, these architectures are growing in complexity, demanding significant storage capacity and computational power, which restricts their deployment on resource-constrained devices. Hence, the need for compressing CNNs before deploying them on such devices.

4 Network pruning definition

Network pruning, inspired by synaptic pruning in the human brain, is one of the most commonly used compression methods. It was initially proposed by Yann LeCun in 1990 [66] to reduce computational and storage requirements, consequently accelerating CNN execution.

The main objective of network pruning is to eliminate parameters that are unnecessary for model prediction or classification and have minimal impact on its performance metrics. Given a CNN model $\mathcal{M}$, the pruning process can be as follows: evaluating the importance of the parameters, selecting and removing the unimportant ones, and then producing a small model $\mathcal{M}'$. Hence, the storage and computation requirements are reduced. Following pruning, the pruned model's feasibility is assessed using criteria such as FLOPs reduction, accuracy impact, parameter reduction, and carbon emission.

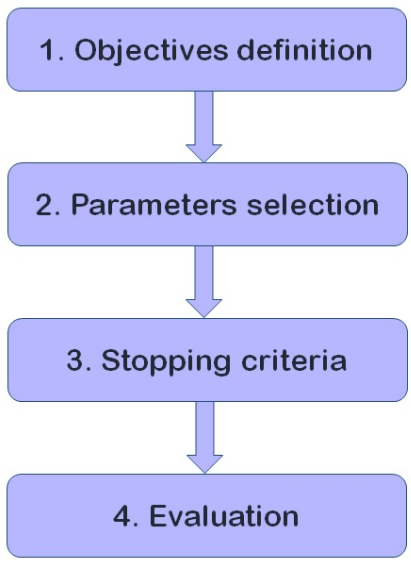

Fig. 4 Basic pruning process

It is possible to derive a general process from most pruning methods. It comprises of the following steps as depicted in Fig. 4: Objectives definition, parameters selection, parameters pruning, and result validation.

4.1 Objective definition

This step can be known as an indicator to specify pruning factors. Therefore, some major questions can be posed, which can be as follows:

- Q1: What type of element can be pruned from the model?
- Q2: When can we prune the model?
- Q3: How can we find the irrelevant parameters?

In this paper, we will address these questions (refer to section 5) and delve into the diverse factors involved in pruning a set of parameters.

4.2 Parameter selection

This step aims to find and prune irrelevant parameters, which is the most critical step in the pruning process due to the complex interconnections among hidden parameters. In this context, numerous methods of searching for irrelevant parameters have been developed. Section 6 goes into greater detail about the classification and descriptions of these methods.

4.3 Stopping criteria

This step serves as an indicator to halt the pruning process. Various stopping criteria are employed, such as when the search is completed, the defined pruning rate is reached, or resource budgets are met.

4.4 Evaluation

The final step involves evaluating the results of the pruning process. This evaluation typically employs various models, datasets, and performance metrics, such as accuracy. Further details on pruning evaluation are provided in Section 7.

5 Key factors of network pruning

This section presents various pruning factors, including descriptions of commonly used ones categorized into three groups: pruning level, pruning time, and pruning rate.

5.1 Pruning level

In this subsection, we will address the question: which elements can we prune from the model? Pruning may occur at any level of the model architecture depending on the pruning objective, which might be memory reduction, computational reduction, or latency reduction. Fig. 5 illustrates the different elements that can be pruned, including weight, node, filter, and layer.

The pruning at the weight level (Fig. 5-(a)), also known as the unstructured level, can obtain a higher compression rate by removing individual connections that are less sensitive to network performance. Many previous studies have pruned the model at the weight level, such as [39, 40, 55, 66, 73]. Despite the memory cost reduction, weight pruning cannot directly speed up inference. On the one hand, the sparse model obtained is inefficient for parallel computing and requires specialized hardware and software to leverage sparse computation and achieve an efficient speed-up during inference [39].

However, node, filter, and layer levels are considered structured levels. They attempt to prune the model in a group to save its structure. Node-level pruning, as shown in Fig. 5 (b), primarily compresses the model by removing less significant nodes in fully connected layers, which have minimal impact on the model's performance (accuracy) [23, 107, 153].

At the filter level, as shown in Fig. 5-(c), the process involves removing irrelevant filters, along with their connections and associated feature maps, from the convolutional layers. It can achieve a significant reduction in computational costs [44, 46, 72, 164].

At the layer level, the entire layer can be pruned to reduce the model's depth, as demonstrated in [31, 56, 148], achieving significant latency reduction.

For even greater compression, pruning can be performed at multiple levels. Chang et al

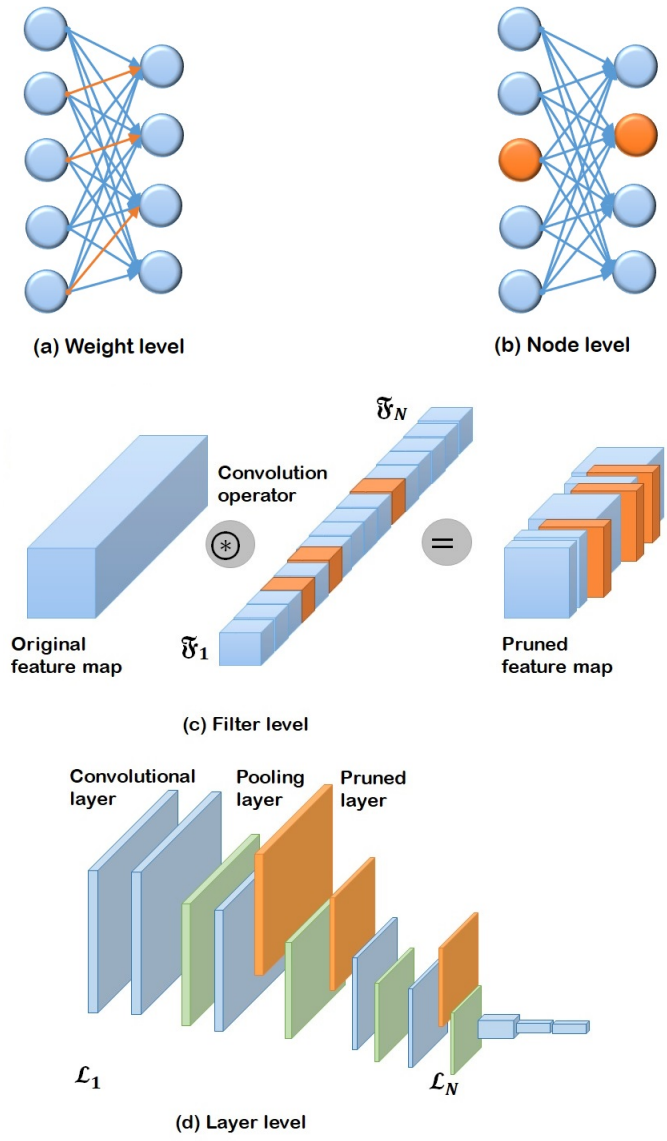

Fig. 5 Network pruning level

[19], Yang et al [159] proposed combining weight pruning and filter pruning techniques.

Table 1 displays the various pruning levels along with their respective advantages and disadvantages.

5.2 Pruning time: when can be applied?

Another critical factor in pruning is determining when it is applied, which involves determining the stage of model development at which pruning occurs.

Pruning in CNN development occurs in three stages: before training, during training, and after training, as depicted in Fig. 6.

Table 2 summarizes the different pruning locations.

5.2.1 Pruning before training

As shown in Fig. 6, pruning can be carried out during the model's initialization phase before

Table 1 Summary of the different pruning levels

Level	Main characteristics	Advantages	Disadvantages
Weight level	Prune the irrelevant weights across layers	-High memory cost reduction -Less sensitive to accuracy degradation	-Unstructured sparse model -Unable to directly speedup the inference
Node level	Prune the irrelevant nodes from the fully connected layers	-High computational cost reduction -Reserve the model structure	-Limited compression ratio -Limited latency reduction
Filter level	Prune the irrelevant filters from the convolutional layers	-High computational cost reduction -Reserve the model structure	-Limited compression ratio -Limited latency reduction
Layer level	Prune the irrelevant layers	-Reserve the model structure -Best result in latency reduction	-Limited compression ratio -High accuracy degradation

training, a technique employed in studies such as [41, 70, 71, 89, 105, 146]. After pruning, the pruned network undergoes standard training. Pruning before training is suitable for reducing training time since it eliminates the need for the entire training process by training only sparse models, making it feasible with limited resources.

However, despite its benefits, pruning before training has the following drawbacks:

- A specific initialization is necessary for successful pruning and training.
- It cannot be used for layer-level pruning because pruning the layer before training renders the model untrained.

5.2.2 Pruning during training

Fig. 6 illustrates the process of pruning during training, which involves identifying and eliminating irrelevant parameters by modifying the training process, as demonstrated in studies like [4, 44, 46, 102, 123, 166].

Pruning during training reduces the need for extensive fine-tuning but can increase the training cost due to modifications in the training process. Additionally, there is an additional energy cost associated with retraining the model when a pre-trained one is already available.

5.2.3 Pruning after training

Conventional pruning algorithms, such as in [40, 72, 101, 135], occur after training. Fig. 6 depicts the pipeline of pruning after training, which commonly has three main phases:

- Training: Begin by training a model until convergence (pre-trained models are sometimes available).
- Pruning: Identify and remove unnecessary parameters.
- Fine-tuning (retraining): Retrain the pruned model to regain the lost accuracy.

Pruning at this stage is appropriate when pre-trained models are available, and there's a need to reduce their inference time. However, when working with an untrained model, it's necessary to train it until convergence as a preprocessing step, in addition to the subsequent retraining phase. This process may result in high resource requirements. Furthermore, while pruning after training can improve inference efficiency, it does not enhance training efficiency.

5.3 Pruning rate

Another critical factor in pruning is the pruning rate, which denotes the percentage of parameters to be removed from the model. The pruning rate can be determined manually or automatically. Table 3 summarizes the various pruning rate strategies.

5.3.1 Handcrafted strategy

Several methods [13, 21, 44, 72, 80, 109, 128, 160] adopt a predefined pruned architecture in which humans indicate how many parameters should be pruned in each layer or a threshold that is used to determine which parameters to remove. Thus, parameters and thresholds were referred to as hyperparameters. Therefore, it requires human expertise to design and decide suitable hyperparameters, which might be difficult and result in sub-optimal pruning.

Table 2 Summary of the different pruning time

When pruning?	Main characteristics	Advantages	Disadvantages
After training	Prune the model after training	-Achieve the reduction of inference time - Best strategy when we have pre-trained model	-Cannot reduce the cost of training -Additional resource to fine-tune the model
During training	Prune the model during training	- Achieve the reduction of inference time -Avoid the need of fine-tuning step	-Cannot reduce the cost of training -Require retraining the model in case we have a pre-trained model
Before training	Prune the model before training	-Achieve the reduction of training and inference time	-Cannot be applied to layer level pruning -Require a specific initialization

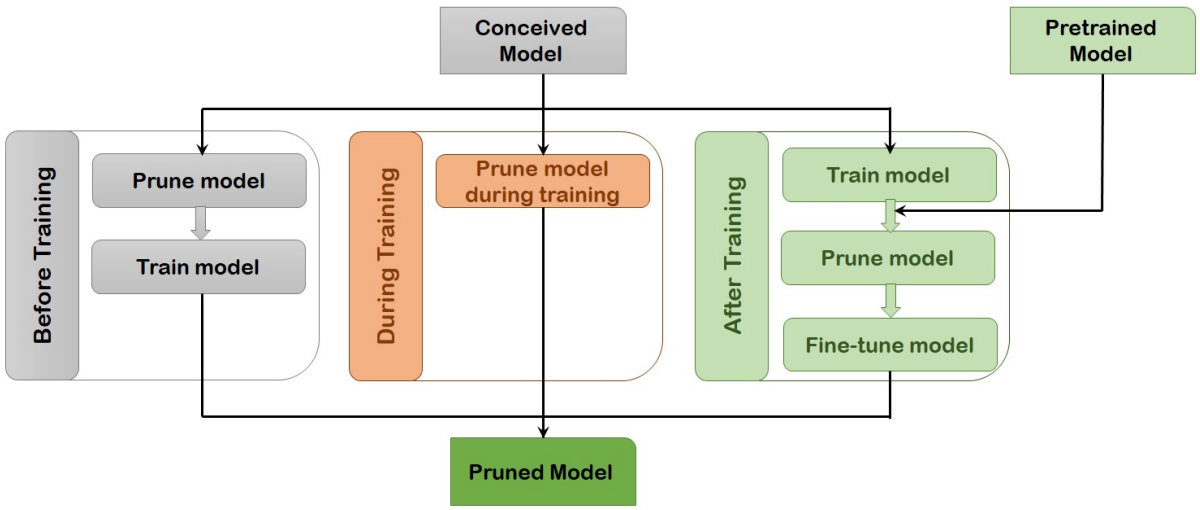

Fig. 6 Pruning and training

5.3.2 Automatic pruning rate

To avoid human intervention, another type of method adopts a fully automated structure in which the pruning rate is automatically determined. For example, [12, 43, 45, 79] automatically find the number of parameters that should be pruned from the model. In the same case, Manessi *et al.* [106] automatically determine the threshold. Consequently, we do not require such input, thus reducing the number of hyperparameters.

6 Pruning units selection method

This section explores the selection methods utilized in pruning research to identify irrelevant parameters. As illustrated in Fig. 7, these methods are typically classified into three categories: criteria-based methods (subsection 6.1), embedded methods (subsection 6.2), and automatic methods (subsection 6.3). Table 4 provides a concise overview of these parameter selection methods.

6.1 Criteria-based method

The criteria-based method stands out as one of the earliest and most commonly employed techniques in pruning research. This approach involves assessing the importance of parameters using predefined criteria, ranking them based on their scores, and subsequently removing those with lower scores. One of its major advantages is its typically lower computational complexity, which makes it well-suited for addressing high-dimensional network pruning challenges. The criteria used for assessment can be categorized into

Table 3 Summary of the different pruning rate strategies

Strategy	Main characteristics	Advantages	Disadvantages
Predefined	The pruning rate is defined by expert	Requires little computation consumption.	Requires human experts to define the pruning rate May produce the sub-optimal pruning
Automatic	The pruning rate is automatically defined	Avoid the sub-optimal solution and the human intervention	Requires more time and computation consuming

Table 4 Summary of the different pruning units selection methods

Method	Main characteristics	Advantages	Limits
Criteria-based method	Select the irrelevant parameters using a predefined criteria	-Simple method -Low complexity	Require manual efforts for designing appropriate criteria -Fall into sub-optimal solution
Embedded method	Embed the parameter selection into the model loss function	-Simultaneously perform parameters selection and model training	-Computationally cost to convergence
Automatic method	Select the irrelevant parameters using learning algorithms	-Optimal result -Save human effort	-High computational requirements

two main groups: data-independent criteria and data-dependent criteria.

6.1.1 Data-independent based criteria

This type is related to the weight of the parameters to measure their importance. There are three categories of data-independent criteria: magnitude-based criteria, correlation-based criteria, and sensitivity-based criteria.

- **Magnitude-based criteria** are selection criteria based on the magnitude of the parameter weight. Therefore, a parameter with a small weight value is considered unimportant than a parameter with a high weight value. Various criteria have been proposed based on the magnitude of the weights. For example, Han *et al.* [40] used the absolute value as a criterion for weight pruning and pruned the lowest value. On the other hand, Li *et al.* (2017) [72] used the L1 norm, and He *et al.* [44] used the L2 norm as criteria for filter pruning, and they pruned the filters with the lowest norm value. However, these criteria have some limitations. They treat the parameters independently, ignoring the possibility of redundancy between high-magnitude parameters. Therefore, they cannot reduce model redundancy.
- **Correlation-based criteria** measure the importance of parameters based on their correlation. Parameters containing redundant information are removed from the model. Some of the commonly used correlation criteria presented in the literature are Geometric median [46]), Cosine distances [8], the distance between filters [169], and Pearson correlation coefficient [132].
- **Sensitivity-based criteria** determine the importance of each parameter by measuring its impact on the final precision. Therefore, the parameters with less impact are removed [22, 155].

6.1.2 Data-dependent based criteria

This type of criteria considers the information in the activation map generated by the parameters to measure their importance.

- **Class label-independent criteria** calculate the importance of the parameters directly from their activation maps without considering the class labels. Hu *et al.* 2016 [49] calculated the average percentage of zero (APoZ) in each activation map and pruned the filters having more zeros in their activation map. Luo and Wu [99] pruned the filters based on the entropy of the activation map. Liu *et al.* [85] calculated the mean gradient of the feature map of each channel to identify the less meaningful one then

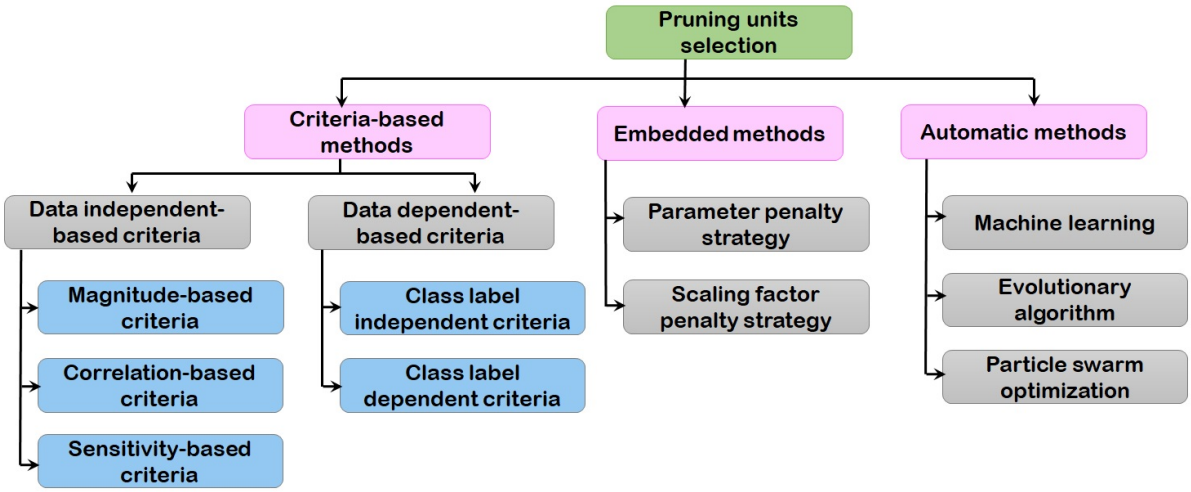

Fig. 7 Taxonomy of Pruning units selection methods

the corresponding channel will be removed. On the other hand, Lin *et al.* [78] removed filters based on the rank of their activation map. Li *et al.* [74] used the mean standard deviation and cosine similarity to prune similar feature maps and their corresponding filters. Wang *et al.* [149] introduce a new theory, Quantified Similarity of Feature Maps (QSFM), to identify redundant information in three-dimensional tensors. [158] pruned the irrelevant filters based on feature channel similarity. In [75], the importance of filters is assessed by measuring the similarity between feature maps using the Hamming distance.

- **Class label-dependent criteria** In their work [125], Sarvani *et al.* determined the importance of filters by utilizing mutual information between the activation map and the class labels as the criterion.

6.1.3 Discussion

Despite its simplicity and low time complexity, the criteria-based method exhibits poor stability. It often requires human expertise to design criteria that are effective across all networks, leading to suboptimal solutions that limit acceleration and compression ratios. Moreover, studies based on this method often apply a single criterion to different layers with varied functions and parameter distributions, neglecting the potential benefits of other criteria, which can impact the results. To address this challenge, He et al. [47] introduced Learning Filter Pruning Criteria (LFPC), which dynamically selects suitable pruning criteria for different functional layers, thereby accounting for their distinctions. Similarly, Pattanayak et al. [112] proposed a novel pruning technique named CURATING, which identifies and retains filters characterized by low redundancy, high saliency, and the generation of high activations. Table 5 provides a high-level summary of criteria-based methods.

6.2 Embedded method (sparsity learning)

The embedded method incorporates the search for irrelevant parameters into the training process, aiming to conduct parameter selection and neural network training simultaneously. This approach involves augmenting the loss function with a penalty term to promote model sparsity without directly assessing parameter importance. Typically, the modified loss function takes the form:

$$loss = l(f(x, W), y) + g(m_i) \tag{1}$$

Where l(.) is the original loss function of the model such as mean-squared error for regression or cross-entropy loss for classification, g(.) is the penalty function on m_i that can be the model weights or scaling factors. W denotes the trainable weights, and (x,y) is the used training set.

Table 5 Summary of criteria-based methods

Year	Reference	Level	Criteria	When?	Pruning rate
2015	Han et al [40]	Weight	Weight magnitude	After training	Predefined
2016	Hu et al [49]	Neuron	APoZ	After training	Predefined
2017	Luo and Wu [99]	Filter	Entropy	After training	Predefined
2017	Li et al [72]	Filter	L1 norm	After training	Predefined
2019	He et al [46]	Filter	Geometric median	During training	Predefined
2019	He et al [44]	Filter	L2 norm	During training	Predefined
2019	Ayinde et al [8]	Filter	Cosine similarity	After training	Predefined
2019	Liu and Wu [85]	Channel	Mean gradient	After training	Predefined
2020	Singh et al [132]	Filter	Pearson Correlation coefficient	After training	Automatic
2020	Chen et al [22]	Channel	Sensitivity	After training	Automatic
2020	Lin et al [78]	Filter	Rank feature map	After training	Predefined
2020	Li et al [74]	Filter	MSD + MCS	After training	Predefined
2021	Wang et al [149]	Filter	QSFM	After training	Predefined
2022	Sarvani et al [125]	Filter	Mutual information	After training	Predefined
2023	Li et al [75]	Filter	Hamming's distance	After training	Predefined
2023	Zhang et al [168]	Filter	Discrete cosine transform	After training	Predefined
2023	Ghimire *et al.* [35]	Filter	L_p norm & Euclidean distance	After training	Automatic
2023	Liu et al [93]	Filter	2D entropy+Feature similarity	After Training	Predefined
2024	Dong et al [29]	Channel	Jensen-Shannon divergence	After training	Predefined
2024	Yang et al [157]	Filter	SFI criterion	During training	Predefined

The following is an introduction to penalty strategies.

6.2.1 Parameter penalty strategy

The penalty function is directly applied to the model's parameters to push the irrelevant ones to zero or near zero during training. After training, the parameters with zero or near-zero values are pruned. Various penalty functions are applied to parameters such as the L1 norm with the capped L1 norm [64], modified $L1/2$ penalty [16], group $L1/2$ regularization [5], group lasso regularization [150], incorporated L2,1 and L2,0 [82], transformed L_1 regularization [103].

6.2.2 Scaling-factor penalty strategy

This strategy focuses on applying the penalty functions to the scaling factors, where each one is associated with a specific structure. These factors may be the factors of the batch-normalization layer [95, 170], or extra ones [53, 83, 115, 152, 156, 167]. This strategy consists of two steps: 1) during training, forcing the factors to be zero; and 2) after training, pruning the parameters related to zero or small factors.

6.2.3 Discussion

The embedded method, despite its simultaneous parameter selection and neural network training, has several drawbacks stemming from alterations to the training process, particularly the modification of the loss function. This method often relies on manually crafted rules and domain expertise, rendering it suboptimal and time-consuming. Moreover, the process of alternating between updating the model's weights and scaling factors, along with computing additional regularization gradients, can significantly increase computational demands and prolong convergence times, particularly when applied to deep CNN models on large datasets. Additionally, since regularization may not precisely drive values to zero using gradient descent, this method often requires further refinement through human expertise and hyperparameter analysis. For instance, some authors established a predefined threshold and subsequently pruned parameters when their values [5, 16] or associated factors [95] were below this threshold. On the other hand, Lin et al [83] and Huang and Wang [53] used FISTA and accelerated Proximal Gradient, respectively, as specialized optimizers to update the scaling factors and give them a binary value. The parameters with zero scaling factor will be pruned. Table 6 provides a high-level summary of embedded methods.

Table 6 Summary of embedded methods

Year	Reference	Level	Penalty imposed on	Regularization	Optimizer
2016	Wen et al [150]	Filter	Model weights	Group lasso	SGD
2017	Liu et al [95]	Channel	Scalar factor of BN	L_1 regularization	SGD
2018	Chang and Sha [16]	weight	Model weights	Modified $L_{1/2}$	SGD
2018	Huang and Wang [53]	Filter	Extra scaling factor	L_1 regularization	APG
2019	Alemu et al [5]	Node	Model weights	Group $L_{1/2}$	SGD
2019	Lin et al [82]	Filter	Model weights	$L_{2,1}$ and $L_{2,0}$	AULM
2019	Ma et al [103]	Weight	Model weights	Transformed L_1	SGD
2019	Lin et al [83]	Filter	Extra scaling factor	L_1 regularization	GAL
2019	Yang et al [156]	Filter	Extra scaling factor	L_1 regularization	SGD
2019	Xiao et al [152]	Weight	Extra scaling factor	L_1regularization	SGD
2020	Ramakrishnan et al [115]	Filter	Extra scaling factor	L_1 regularization	SGD
2022	Tang et al [139]	Weight	Extra scaling factor	L_2 regularization	SGD
2023	Zhang and Freris [170]	Filter	Scalar factor of BN	L_1-regularized	Proximal gradient
2024	Zhang et al [167]	multi-level	Extra scaling factor	L_1 regularization	SGD

6.3 Automatic based methods

Automatic-based methods utilize learning algorithms to identify irrelevant parameters. These methods are more efficient than criteria-based and embedded as they employ fully automated pruning techniques, which can enhance pruning quality by generating optimal parameter selections and reducing human effort.

These methods leverage various learning algorithms to achieve their objectives. A summary of automatic-based methods for parameter selection, derived from the reviewed articles, is presented in Table 7.

6.3.1 Machine-learning algorithms

- **Reinforcement learning** Huang et al [52] proposed a method based on reinforcement learning (RL) to evaluate the importance of the parameters. They train pruning agents to search and prune unnecessary channels layer by layer from the pretrained model, followed by fine-tuning.
- **Deep neural networks** Li *et al.* [76] used the DNN model to search for irrelevant filters. Specifically, the DNN extracts the features from the filters. Then, they employed the k-means algorithm to cluster features into groups and mapped the clustering results to the filters to determine their similarity. Finally, they retain the closest filter to the centroid in each cluster and prune the others. On the other hand, Verma *et al.* [144] proposed a multitask network to give weight to each feature map and try to assign zero value to this weight. Then, the feature map and corresponding filter are pruned if their corresponding weight value is zero. However, the proposed DNN-based approaches usually require a long time for training and learning to find the irrelevant parameters.
- **Layered approaches** is a parameter selection method based on an efficient independent pruning layer to find less salient filters during fine-tuning [20, 58, 100, 140]. This layer takes the activation responses of the convolutional layer as input and generates a binary code. After training, the filters with the corresponding '0' code are removed.

The drawback of these proposed methods is to search the weak parameters layer by layer and are based on the learned weights to judge their importance. On the other hand, Lui *et al.* [96] show that the learned weights are unimportant in the pruning process and conjecture that the essence of parameter pruning is searching for good optimal substructure rather than iteratively searching salient parameters.

6.3.2 Evolutionary algorithms

Recently, Evolutionary algorithms (EA) have been adopted to perform network pruning, to automatically search for the optimal pruned network.

- **Evolution strategy (ES):** Fernandes Jr and Yen [34] used the evolution strategy for pruning

Table 7 Summary of automatic methods

Year	Reference	Level	Search method
2018	Huang et al [52]	Channel	Reinforcement learning
2019	Li et al [76]	Filter	Deep learning
2019	Zhou et al [176]	Filter	Multiobjective evolutionary algorithm
2019	Zhou et al [175]	Filter	Multiobjective evolutionary algorithm
2019	Kim et al [58]	Channel	Trainable pruning layer
2019	Chen et al [20]	Channel	Saliency-and-Pruning Module
2019	Liu et al [97]	Filter	Evolution strategy algorithm
2020	Zhang et al [172]	Filter and layer	Evolutionary algorithm
2020	Verma et al [144]	Filter	Multitask DNN
2020	Luo and Wu [100]	Filter	Trainable pruning layer
2021	Tian et al [140]	Filter	Collaborative layers
2021	Fernandes Jr and Yen [34]	Filter	Multiobjective Evolutionary algorithm
2021	Zhang et al [171]	Filter	Multiobjective evolutionary algorithm
2021	Tmamna et al [141]	Layer	Improved PSO
2021	Zhou et al [177]	Layer	Multiobjective evolutionary algorithm
2022	Chang et al [17]	Filter	Clustering and PSO
2022	Skandha et al [133]	Filter	Genetic Algorithm (GA)
2022	Shang et al [127]	Filter	Cooperative CoEvolution algorithm (CCE)
2023	Liu et al [94]	Filter	Social Group Optimization (SGO) algorithm
2023	Agarwal et al [2]	Filter	Particle Swarm Optimization (PSO) algorithm
2023	Poyatos et al [113]	Neuron	Genetic Algorithm
2024	Xu et al [154]	Filter	Artificial bee colony algorithm
2024	Liang et al [77]	Channel	Artificial bee colony algorithm

to search for the optimal filter set. ES algorithm initializes a population of individuals as an identical copy of the original CNN model being pruned. Then, each individual is mutated to generate a pruned version of the original model.

- **Genetic algorithm (GA):** is adopted as an optimizer to solve a single objective [97, 172]. Furthermore, some work [171, 175, 176] treats network pruning as a multi-objective optimization problem and uses EA to find an optimal pruned network that strikes a good balance between network size and performance. In [133], the authors successfully compressed CNN models using a genetic algorithm for the classification of lung diseases. Their findings showed that on the LIDC-IDRI lung dataset, the proposed CNN model could be reduced in size by 90.3% while maintaining its performance.
- **Cooperative CoEvolution algorithm:** Shang et al [127] adopted the Cooperative CoEvolution algorithm (CCE) for filter Pruning. First, the network is split into multiple groups, then a separate EA is adopted for each group.
- **Particle swarm algorithm (PSO):** is also used to search for the optimal pruned. For instance, Agarwal *et al.* [2] used a conventional PSO to develop a pruned version of VGG16-based Fully Convolution Network (FCN). Tmamna et al [141] proposed an improved PSO to perform layer pruning. Chang et al [17] further proposed channel pruning based on clustering and PSO (ACP). Here, clustering via the similarity of the feature maps is adopted to perform preliminary pruning on the network. Then conventional PSO is adopted to find the optimal pruned network. The proposed method achieved a reduction of 73.44% FLOPs with a 0.15% accuracy drop for VGG16.
- **Social Group Optimization (SGO):** In [94], Liu *et al.* used the Social Group Optimization (SGO) algorithm to search for the optimal pruned structure. They introduced the k-means++ method to hierarchically cluster filters with similar features in each convolutional layer, forming an initial compact compression structure. Subsequently, they used the SGO algorithm to search and optimize the compression process of the post-clustered structure,

ultimately identifying the optimal compressed structure.

- **Artificial bee colony (ABC) algorithm:** Liang et al [77] applied the ABC algorithm to optimize CNN models by selectively removing channels thereby enhancing the speed of human activity inference on mobile devices. Xu et al [154] introduced Filter Pruning via Adaptive Automatic Structure Search (FP-AASS), considering filter pruning as an optimization task. FP-AASS utilizes the ABC algorithm to automatically search for the optimal pruned network meeting parameters and FLOPs constraints.

6.3.3 Discussion

Even though AutoML methods perform well with parameter selection, they have several drawbacks that do not work well with large deep models. Therefore, it is very computationally expensive to search for irrelevant parameters, which imposes additional computational overhead on top of the computational complexity of the training. For example, Huang et al [52] not only requires a training model but also trains RL agents, which is very costly.

Overall, CNN pruning methods are commonly categorized into three groups: criteria-based methods, embedded methods, and automatic methods based on the approach used to detect irrelevant parameters. This taxonomy remains applicable and adaptable to other DNN architectures like transformers. For instance, Tmamna et al [142] presented an automatic method using BPSO to identify irrelevant units in vision transformer (ViT). Furthermore, Yu and Xiang [165] introduced an embedded method for removing irrelevant units. Their method utilizes an explainability-aware mask to assess each unit's contribution to predicting each class, which is fully differentiable and learned with a class-wise regularizer.

7 Pruning evaluation

To evaluate the performance of any proposed pruning method, the presence of standard datasets and models is essential. These standards facilitate the comparison of the performance between the original model and the pruned one. Performance evaluation can be carried out in many ways, depending on the datasets and the model used. The results of this validation can determine the efficacy of the pruning methods. The dataset, the models, and the performance measures referred to in the literature are described below.

7.1 Datasets

We report the most common datasets used for evaluation in the literature.

- **CIFAR dataset** consists of 32×32 RGB images, where the images are devised as 50 K for training and 10K for testing [62]. It includes two categories, which are CIFAR10 describing 10 classes of objects, and CIFAR100 describing 100 classes of objects.
- **ImageNet dataset** is a large-scale image classification dataset [124] containing 224×224 RGB images. It comprises 1.2 million images for training and 50,000 for testing, covering 1,000 object classes.
- **Tiny ImageNet** is a subset of ImageNet, comprises 100,000 training images and 10,000 test images, representing 200 classes and downsized to 64×64 colored images [65].
- **MNIST** is a large database of handwritten digits, which consists of 60,000 28x28 grayscale images of the 10 classes, along with 10,000 test images [67].
- **CUB-200 dataset** contains 11788 images of 200 classes which are divided into 5994 images for training and 5794 for test [145].
- **Indoor-67 dataset** contains 6700 images of 67 classes split 5360 training and 1340 test images [114].
- **SVHN dataset**, called Street View House Number (SVHN) dataset, has over 60,0000 32×32 colored digit images (73257/26032/531131 images for training/testing/additional) of 10 classes [110].
- **Microsoft COCO** is a vast object detection dataset. It comprises 80 categories. The training and validation sets for the yearly competition encompass 120,000 images, while the test set consists of over 40,000 images [84].

The CIFAR and ImageNet datasets are the most popular among these datasets because there

are numerous pre-trained models available for these two datasets.

7.2 Models

This section introduces the common models used by the existing works to evaluate their methods. Parameters and pruning levels of different models are summarized in Table 8.

- **LeNet** was developed in the year 1998 by Yann LeCun *et al.* [67]. The basic LeNet 5 architecture consists of two convolutional layers, two sub-sampling layers, two fully connected layers, and an output layer with the Gaussian connection.
- **AlexNet** was developed by Alex Krizhevesky and others in 2012 [63]. The architecture of AlexNet comprises eight layers, including five convolutional layers, two fully connected layers, and a softmax layer at the end.
- **VGGNet** was developed by Karen Simonyan and stood for Visual Geometry Group [131]. This model has three variants, namely VGG11, VGG16, and VGG19 which differ only in the total number of layers in the network.
- **GoogLeNet** was developed by Christian Szegedy of Google in the year 2014 with the intent to reduce the computational complexity compared to the traditional model [136]. The architecture is 22 layers deep, based on the Inception architecture.
- **ResNet**, called Residual Neural Network, was introduced by Kaiming He *et al.* [42] in 2015 to design ultra-deep networks that mitigate the vanishing gradient problem. The architecture of ResNet consists of several residual blocks. Several variants of ResNet such as ResNet-18, ResNet-20, ResNet-32, ResNet-44, ResNet-50, ResNet-56, ResNet-110, and ResNet-152, have been proposed, differing in the number of layers which can exceed 100 or even 1000 layers.
- **DenseNet** is a CNN architecture with dense connections proposed by [51]. DenseNet comprises fundamental modules known as Dense Blocks, where each Dense Block takes the feature maps from previous Dense Blocks as input. A feedforward connection is maintained between each layer of DenseNet, thereby strengthening the effect of deep convolution across layers. Several variants of DenseNet are proposed such as DenseNet-40, and DenseNet-121.
- **MobileNet** is a lightweight network developed by Howard *et al.* [48] for use in mobile and embedded vision applications. The architecture of MobileNet is built on depthwise separable convolutions, and only the first layer is a full convolutional layer. Several variants of MobileNet are proposed such as MobileNet-V1, MobileNet-V2, and MobileNet-V3.
- **EfficientNet**, introduced by Tan and Le [137], presents a CNN architecture and scaling method. It uniformly scales all dimensions of depth, width, and resolution using a compound coefficient. Its architecture integrates Mobile Inverted Bottleneck (MBConv) layers, combining depth-wise separable convolutions with inverted residual blocks. Furthermore, it leverages the Squeeze-and-Excitation (SE) optimization to improve its performance.
- **YOLO**, called You Only Look Once, was introduced by Redmon et al [119] to object detection. Several variants of YOLO have been proposed such as YOLOv4, YOLOv5s, and YOLOv7.
- **Unet** is a CNN architecture used for semantic segmentation. It was intorduced by Ronneberger et al [122].

It should be noted that the most frequently used models in the literature are VGGNet and ResNet. However, there is a lack of pruning results on DenseNet, MobileNet, and GoogleNet.

7.3 Performance measures

The metrics used to evaluate the pruned models are as follows.

- **Accuracy** refers to how well a model performs on a given task. It is typically calculated as a measure of how well a model correctly identifies both positive and negative instances. It is calculated as follows:

$$accuracy = \frac{TP + TF}{N} \tag{2}$$

  where TP is the number of true positives, TN is the number of true negatives, and N is the total number of instances.
- **Parameters number** used to evaluate the model's resource requirements. The number of parameters directly contributes to the size of

Table 8 Summary of the different models

Architecture	Model	Parameters (M)	Level pruning
Basic network architecture	LeNet	0.43	Weight/Filter/Neuron
	AlexNet	61.00	Weight/Filter
	VGG-16	138,35	Weight/Filter/Neuron/Layer
	VGG-19	143,66	Weight/Filter/Neuron/Layer
Residual connection architecture	ResNet-18	11.69	Weight/Filter
	ResNet-34	21.90	Weight/Filter
	ResNet-56	0.85	Weight/Filter/Block
	ResNet-110	1.73	Weight/filter/Block
	ResNet 152	60.19	Weight/Filter/Block
Dense connection architecture	DenseNet-40	1.04	Weight/Filter
	DenseNet-121	8.06	Weight/Filter
Depthwise separable convolution	MobileNet-V1	4.20	Weight/Filter
	MobileNet-V2	3.40	Weight/Filter
	MobileNet-V3	5.40	Weight/Filter
Inception architecture	GoogleNet	6.15	Weight/Filter/Branch
EfficientNet architecture	EfficientNet-B0	5.3M	Weight/Filter
	EfficientNet-B1	7.8M	Weight/Filter
YOLO architecture	YOLOv4	63M	Weight/Filter
	YOLOv5s	7.07M	Weight/Filter
	YOLOv7	37.2M	Weight/Filter
UNet architecture	UNet	32M	Weight/Filter

the model. It represents the learnable weights and biases that the model uses to make predictions based on the input data.

- **Floating point operations (FLOPs)** are a metric used to quantify the arithmetic operation (additions, subtractions, multiplications, and divisions) within a model, providing a measure of its computational complexity. They are calculated as follows:

$$FLOPs = 2HW(Cin * k^2 + 1)Cout \quad (3)$$

where Cin and Cout are the numbers of input channels and output channels, K is the kernel width and W, H are the width and height of the output feature map.

- **Latency** refers to the time it takes for a model to process an input and generate a prediction when deployed on specific hardware.
- **Carbon emission** evaluates the amount of carbon dioxide (CO2) emitted during model inference. This metric is used to evaluate the environmental impact of models.
- **Energy consumption** assesses the total amount of energy used during inference, measured in kilowatt-hours (kWh).

For a thorough performance evaluation of pruned models, it's crucial to delve into additional techniques, including uncertainty quantification, sensitivity analysis, and adversarial attacks

- **Uncertainty quantification** is a crucial process for assessing reliability and confidence in model predictions. Techniques can be categorized into aleatoric and epistemic uncertainties, with epistemic uncertainty arising from inherent ambiguity in model parameters and aleatoric uncertainty exploring data variability. Monte Carlo dropout can be used for parameter perturbation evaluation.
- **Sensitivity analysis** evaluates the impact of input perturbations on a model's output, providing insights into the model's robustness and stability. To evaluate the model's sensitivity to variations in input data, the Variable Perturbation Method (VPM) can be used.
- **Adversarial Attacks** involve introducing visually imperceptible perturbations to clean images to compromise a model's predictive capability. These attacks may access the model's parameters and architecture design, termed white-box attacks, or solely target the

model output, known as black-box attacks. Fast gradient sign method (FGSM) and Projected gradient descent (PGD) can be used as attack methods.

8 Future Directions

While neural network pruning has received considerable attention in recent years, it still faces certain limitations. In concluding our paper, we highlight open issues that remain underexplored in the literature and offer promising directions for future research.

8.1 Latency reduction

Many pruning methods have primarily concentrated on reducing model width by eliminating numerous weights and filters, with metrics like FLOPs and parameters used to gauge performance. However, these methods are constrained by model depth and often yield limited latency reduction. This is because latency depends not only on the number of filters per layer but also on the specific deployment device, as noted in [32, 138]. Hence, the primary challenge in the pruning process remains the discovery of pruned models that strike a delicate balance between system performance and the variable resource constraints of specific devices. As a result, more study is required to create techniques that effectively prune the model while taking into account the significance of parameters and the underlying hardware architecture.

8.2 Comparison

A critical observation from our survey is the lack of comprehensive method comparison among various pruning techniques. This absence complicates the process of selecting the most suitable pruning method for specific applications. Furthermore, the absence of standardized benchmarks and metrics poses challenges in comparing pruning methods and assessing the progress made in this field over the years.

For instance, while some studies evaluate their methods solely on architectures like VGG16 or AlexNet, these benchmarks may not be universally applicable to other architectures such as ResNet or MobileNet. Consequently, it becomes challenging to gauge the relative effectiveness of pruning methods across different CNN architectures. To address this issue and facilitate fair and direct comparisons between pruning methods, it is imperative to establish a commonly accepted baseline that incorporates modern architectures such as MobileNet. This standardization would provide a more comprehensive and unbiased evaluation framework for assessing the performance of pruning techniques.

8.3 Synergistic combination of pruning Methods

Since each pruning strategy and method has its weaknesses and strengths, an interesting and relatively unexplored question revolves around how these methods can be synergistically combined to harness their strengths and mitigate their weaknesses. Therefore, Haider and Taj [38] combined width pruning with depth pruning to achieve more compression in terms of latency and memory. Additionally, Louati et al [98] introduced an approach that combines filter and channel pruning methods based on Evolutionary Algorithms (EA). This method involves eliminating filters and channels to decrease the number of parameters and computational complexity of the model. Similarly, Zhang et al [167] introduced the Multi-Granularity Pruning Framework to achieve various levels of pruning granularity, including weight pruning, channel pruning, and filter pruning. On the other hand, Chang [15] combined the criteria-based method and the automatic method to reduce the high computational requirements of the automatic method and the high hyperparameter fine-tuning of the criteria-based method.

We believe there is fertile ground for research investigating such combinations.

8.4 Specific Network Pruning

Besides the prevailing image classification CNNs, pruning is beneficial for other computer vision tasks including object detection, image segmentation, and natural language processing (NLP). Besides, there are some emerging directions to prune specific networks such as:

- Artificial general intelligence (AGI): foundation models such as GPT-3 [11] and generalist agents like the generalist agent Gato [120] are possible

approaches to AGI. These massive models can benefit from pruning research to attain greater efficiency.
- Generative adversarial networks (GAN) [130]: consist of both a generative network and a discriminative network, and these architectures often require significant FLOPs and storage.
- Vision transformers [30]: which achieved better performance in computer vision challenges. Thus, it is meaningful to adopt pruning methods to compress these new architectures.

There is a need for optimizing tradeoffs between performance measures when pruning deep neural networks and comparing the effectiveness of pruning methods using tradeoffs. It is worth raising the lack of pruning methods for some models such as LSTMs, and hybrid CNN-LSTM architectures. The proposed model in [36] is an example of a hybrid architecture that uses both CNNs and LSTMs for audiovisual speech enhancement for which pruning methods need to be developed. Another model was proposed in Passos et al [111] on energy-efficient graph neural network approaches for which new pruning methods also need to be developed to reduce their latency in addition to their existing low energy efficiency consumption benefits. This is required for practical utilization of such deep neural network models in future, latency and energy-constrained multimodal hearing assistive technologies [1]. In the same direction, ConvLSTM models were proposed to handle spatio-temporel information in different applications such as EEG-based emotion recognition [60] and EEG-based epilepsy prediction [61]. While it is very efficient in terms of performance, it contains a very high number of parameters as well as an important value of FLOPs. Thus, it is necessary to prune ConvLSTM models.

9 Conclusion

With the increasing size of CNNs and a growing interest in deploying DL models on edge/IoT devices, there is a pressing need for best practices and techniques to enable the development of efficient and compact models. Pruning emerges as a highly effective approach for accelerating computations and reducing energy consumption on small devices. The essence of model pruning lies in creating a compact model from an existing, over-parameterized, and larger model. This paper explores various strategies and methods for pruning neural networks. We classified the criteria into data dependent and data independent and the pruning level into an unstructured level and structured level. We also covered the different factors that can influence the performance and efficiency of the pruning approach. Finally, the surveyed selection method includes a handcrafted criteria-based method that selects the irrelevant parameters based on predefined criteria, an embedded method that embeds a penalty term into the loss function to decide on the importance of the parameter, and an automatic method based on machine learning and evolutionary algorithms to search the unimportant parameters.

we outline several open challenges that persist, encompassing not only the pursuit of improved performance but also considerations related to latency, hardware dependencies, benchmarking methodologies, the synergistic combination of pruning methods, and reduced training times. Our survey is tentative to contribute to the continued development by giving an outline of the current state of research and highlighting significant open challenges. An emphasis on the importance of green deep learning is made to further recall the negative effects of using large models on our environment and our globe.

Authors' Contribution

Jihene Tmamna: Conceptualization, Methodology, Software, Writing - original draft.
Emna Ben Ayed: Conceptualization, Methodology, Software, Writing - original draft.
Rahma Fourati: Conceptualization, Methodology, Software, Writing - original draft.
Mandar Gogate: Investigation, Writing - review & editing.
Tughrul Arslan: Investigation, Writing - review & editing.
Amir Hussain: Conceptualization, Investigation, Project administration.
Mounir Ben Ayed: Conceptualization, Investigation, Project administration.

Funding

The research leading to these results has received funding from the Ministry of Higher Education and Scientific Research of Tunisia under grant agreement number LR11ES48.
Professor Hussain acknowledges the support of the UK Engineering and Physical Sciences Research Council (EPSRC) (Grants No. EP/M026981/1, EP/T021063/1, EP/T024917/1)

Data Availability

The datasets described in this survey are publicly accessible.

Conflict of Interest

The authors declare that they have no conflict of interest.

Compliance with Ethical Standards

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.
Informed Consent: Not applicable.

References

[1] Adeel A, Adetomi A, Ahmed K, et al (2023) Unlocking the potential of two-point cells for energy-efficient and resilient training of deep nets. IEEE Transactions on Emerging Topics in Computational Intelligence 7(3):818–828

[2] Agarwal M, Gupta SK, Biswas K (2023) Development of a compressed fcn architecture for semantic segmentation using particle swarm optimization. Neural Computing and Applications pp 1–14

[3] Ahn S, Hu SX, Damianou A, et al (2019) Variational information distillation for knowledge transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9163–9171

[4] Aketi SA, Roy S, Raghunathan A, et al (2020) Gradual channel pruning while training using feature relevance scores for convolutional neural networks. IEEE Access 8:171,924–171,932

[5] Alemu HZ, Zhao J, Li F, et al (2019) Group $l\_\{1/2\}$ regularization for pruning hidden layer nodes of feedforward neural networks. IEEE Access 7:9540–9557

[6] Allen-Zhu Z, Li Y, Liang Y (2019) Learning and generalization in overparameterized neural networks, going beyond two layers. Advances in neural information processing systems 32

[7] Astrid M, Lee SI (2018) Deep compression of convolutional neural networks with low-rank approximation. ETRI journal 40(4):421–434

[8] Ayinde BO, Inanc T, Zurada JM (2019) Redundant feature pruning for accelerated inference in deep neural networks. Neural Networks 118:148–158

[9] Banner R, Hubara I, Hoffer E, et al (2018) Scalable methods for 8-bit training of neural networks. Advances in neural information processing systems 31

[10] Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:200410934

[11] Brown T, Mann B, Ryder N, et al (2020) Language models are few-shot learners. Advances in neural information processing systems 33:1877–1901

[12] Cai H, Lin J, Lin Y, et al (2019) Automl for architecting efficient and specialized neural networks. IEEE Micro 40(1):75–82

[13] Cai L, An Z, Yang C, et al (2021) Softer pruning, incremental regularization. In: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, pp 224–230

[14] Cai Y, Yao Z, Dong Z, et al (2020) Zeroq: A novel zero shot quantization framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 13,169–13,178

[15] Chang J (2020) Coarse and fine-grained automatic cropping deep convolutional neural network. arXiv preprint arXiv:201006379

[16] Chang J, Sha J (2018) Prune deep neural networks with the modified $l\_\{1/2\}$ penalty. IEEE Access 7:2273–2280

[17] Chang J, Lu Y, Xue P, et al (2022) Automatic channel pruning via clustering and swarm intelligence optimization for cnn. Applied Intelligence pp 1–21

[18] Chang J, Lu Y, Xue P, et al (2023) Iterative clustering pruning for convolutional neural networks. Knowledge-Based Systems 265:110,386

[19] Chang X, Pan H, Lin W, et al (2021) A mixed-pruning based framework for embedded convolutional neural network acceleration. IEEE Transactions on Circuits and Systems I: Regular Papers 68(4):1706–1715

[20] Chen J, Zhu Z, Li C, et al (2019) Self-adaptive network pruning. In: Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part I 26, Springer, pp 175–186

[21] Chen Y, Wen X, Zhang Y, et al (2021) Ccprune: Collaborative channel pruning for learning compact convolutional networks. Neurocomputing 451:35–45

[22] Chen Z, Xu TB, Du C, et al (2020) Dynamical channel pruning by conditional accuracy change for deep neural networks. IEEE transactions on neural networks and learning systems 32(2):799–813

[23] Cheng Y, Yu FX, Feris RS, et al (2015) An exploration of parameter redundancy in deep networks with circulant projections. In: Proceedings of the IEEE international conference on computer vision, pp 2857–2865

[24] Chmiel B, Ben-Uri L, Shkolnik M, et al (2020) Neural gradients are near-lognormal: improved quantized and sparse training. arXiv preprint arXiv:200608173

[25] Collobert R, Weston J, Bottou L, et al (2011) Natural language processing (almost) from scratch. Journal of machine learning research 12(ARTICLE):2493–2537

[26] Cong S, Zhou Y (2023) A review of convolutional neural network architectures and their optimizations. Artificial Intelligence Review 56(3):1905–1969

[27] Denton EL, Zaremba W, Bruna J, et al (2014) Exploiting linear structure within convolutional networks for efficient evaluation. Advances in neural information processing systems 27

[28] Dong X, Yang Y (2019) Network pruning via transformable architecture search. Advances in Neural Information Processing Systems 32

[29] Dong Z, Duan Y, Zhou Y, et al (2024) Weight-adaptive channel pruning for cnns based on closeness-centrality modeling. Applied Intelligence 54(1):201–215

[30] Dosovitskiy A, Beyer L, Kolesnikov A, et al (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:201011929

[31] Elkerdawy S, Elhoushi M, Singh A, et al (2020) One-shot layer-wise accuracy approximation for layer pruning. In: 2020 IEEE International Conference on Image Processing (ICIP), IEEE, pp 2940–2944

[32] Elkerdawy S, Elhoushi M, Singh A, et al (2020) To filter prune, or to layer prune, that is the question. In: Proceedings of the Asian Conference on Computer Vision

[33] Farhadi A, Redmon J (2018) Yolov3: An incremental improvement. In: Computer vision and pattern recognition, Springer Berlin/Heidelberg, Germany, pp 1–6

[34] Fernandes Jr FE, Yen GG (2021) Pruning deep convolutional neural networks architectures with evolution strategy. Information Sciences 552:29–47

[35] Ghimire D, Kim SH (2023) Magnitude and similarity based variable rate filter pruning for efficient convolution neural networks. Applied Sciences 13(1):316

[36] Gogate M, Dashtipour K, Adeel A, et al (2020) Cochleanet: A robust language-independent audio-visual model for real-time speech enhancement. Information Fusion 63:273–285

[37] Hafiz A, Bhat R, Hassaballah M (2023) Image classification using convolutional neural network tree ensembles. Multimedia Tools and Applications 82(5):6867–6884

[38] Haider MU, Taj M (2021) Comprehensive online network pruning via learnable scaling factors. In: 2021 IEEE International Conference on Image Processing (ICIP), IEEE, pp 3557–3561

[39] Han S, Mao H, Dally WJ (2015) Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:151000149

[40] Han S, Pool J, Tran J, et al (2015) Learning both weights and connections for efficient neural network. Advances in neural information processing systems 28

[41] Hayou S, Ton JF, Doucet A, et al (2020) Pruning untrained neural networks: Principles and analysis. arXiv preprint arXiv:200208797

[42] He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

[43] He Y, Han S (2018) Adc: Automated deep compression and acceleration with reinforcement learning. arXiv preprint arXiv:180203494 2

[44] He Y, Kang G, Dong X, et al (2018) Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:180806866

[45] He Y, Lin J, Liu Z, et al (2018) Amc: Automl for model compression and acceleration on mobile devices. In: Proceedings of the European conference on computer vision (ECCV), pp 784–800

[46] He Y, Liu P, Wang Z, et al (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4340–4349

[47] He Y, Ding Y, Liu P, et al (2020) Learning filter pruning criteria for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2009–2018

[48] Howard AG, Zhu M, Chen B, et al (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:170404861

[49] Hu H, Peng R, Tai YW, et al (2016) Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint arXiv:160703250

[50] Hu W, Che Z, Liu N, et al (2023) Channel pruning via class-aware trace ratio optimization. IEEE Transactions on Neural Networks and Learning Systems

[51] Huang G, Liu Z, Van Der Maaten L, et al (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708

[52] Huang Q, Zhou K, You S, et al (2018) Learning to prune filters in convolutional neural networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, pp 709–718

[53] Huang Z, Wang N (2018) Data-driven sparse structure selection for deep neural networks. In: Proceedings of the European conference on computer vision (ECCV), pp 304–320

[54] Jiao X, Yin Y, Shang L, et al (2019) Tinybert: Distilling bert for natural language understanding. arXiv preprint arXiv:190910351

[55] Jin S, Di S, Liang X, et al (2019) Deepsz: A novel framework to compress deep neural networks by using error-bounded lossy compression. In: Proceedings of the 28th international symposium on high-performance parallel and distributed computing, pp 159–170

[56] Jordao A, Lie M, Schwartz WR (2020) Discriminative layer pruning for convolutional neural networks. IEEE Journal of Selected Topics in Signal Processing 14(4):828–837

[57] Kholiavchenko M (2018) Iterative low-rank approximation for cnn compression. arXiv preprint arXiv:180308995

[58] Kim J, Park C, Jung H, et al (2019) Differentiable pruning method for neural networks. CoRR

[59] Kim YD, Park E, Yoo S, et al (2015) Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv preprint arXiv:151106530

[60] Kouka N, Fourati R, Fdhila R, et al (2023) Eeg channel selection-based binary particle swarm optimization with recurrent convolutional autoencoder for emotion recognition. Biomedical Signal Processing and Control 84:104,783

[61] Kouka N, Fourati R, Baghdadi A, et al (2024) A mutual information-based many-objective optimization method for eeg channel selection in the epileptic seizure prediction task. Cognitive Computation pp 1–19

[62] Krizhevsky A, Hinton G, et al (2009) Learning multiple layers of features from tiny images. ” ”

[63] Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems, pp 1106–1114

[64] Kumar A, Shaikh AM, Li Y, et al (2021) Pruning filters with l1-norm and capped l1-norm for cnn compression. Applied Intelligence 51:1152–1160

[65] Le Y, Yang X (2015) Tiny imagenet visual recognition challenge. CS 231N 7(7):3

[66] LeCun Y, Denker J, Solla S (1990) Optimal brain damage. Advances in neural information processing systems 2

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

[68] Lee D, Kwon SJ, Kim B, et al (2019) Learning low-rank approximation for cnns. arXiv preprint arXiv:190510145

[69] Lee J, Yu M, Kwon Y, et al (2022) Quantune: Post-training quantization of convolutional neural networks using extreme gradient boosting for fast deployment. Future Generation Computer Systems 132:124–135

[70] Lee N, Ajanthan T, Torr PH (2018) Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:181002340

[71] Lee N, Ajanthan T, Gould S, et al (2019) A signal propagation perspective for pruning neural networks at initialization. arXiv preprint arXiv:190606307

[72] Li H, Kadav A, Durdanovic I, et al (2016) Pruning filters for efficient convnets. arXiv preprint arXiv:160808710

[73] Li H, Liu N, Ma X, et al (2019) Admm-based weight pruning for real-time deep learning acceleration on mobile devices. In: Proceedings of the 2019 on Great Lakes Symposium on VLSI, pp 501–506

[74] Li H, Ma C, Xu W, et al (2020) Feature statistics guided efficient filter pruning. arXiv preprint arXiv:200512193

[75] Li J, Shao H, Zhai S, et al (2023) A graphical approach for filter pruning by exploring the similarity relation between feature maps. Pattern Recognition Letters 166:69–75

[76] Li L, Zhu J, Sun MT (2019) Deep learning based method for pruning deep neural networks. In: 2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), IEEE, pp 312–317

[77] Liang J, Zhang L, Bu C, et al (2024) An automatic network structure search via channel pruning for accelerating human activity inference on mobile devices. Expert Systems with Applications 238:122,180

[78] Lin M, Ji R, Wang Y, et al (2020) Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1529–1538

[79] Lin M, Ji R, Zhang Y, et al (2020) Channel pruning via automatic structure search. arXiv preprint arXiv:200108565

[80] Lin M, Cao L, Li S, et al (2021) Filter sketch for network pruning. IEEE Transactions on Neural Networks and Learning Systems 33(12):7091–7100

[81] Lin M, Cao L, Zhang Y, et al (2022) Pruning networks with cross-layer ranking & k-reciprocal nearest filters. IEEE Transactions on Neural Networks and Learning Systems

[82] Lin S, Ji R, Li Y, et al (2019) Toward compact convnets via structure-sparsity regularized filter pruning. IEEE transactions on neural networks and learning systems 31(2):574–588

[83] Lin S, Ji R, Yan C, et al (2019) Towards optimal structured cnn pruning via generative adversarial learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2790–2799

[84] Lin TY, Maire M, Belongie S, et al (2014) Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, Springer, pp 740–755

[85] Liu C, Wu H (2019) Channel pruning based on mean gradient for accelerating convolutional neural networks. Signal Processing 156:84–91

[86] Liu H, Simonyan K, Vinyals O, et al (2017) Hierarchical representations for efficient architecture search. arXiv preprint arXiv:171100436

[87] Liu J, Tripathi S, Kurup U, et al (2020) Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey. arXiv preprint arXiv:200504275

[88] Liu S, Lin Y, Zhou Z, et al (2018) On-demand deep model compression for mobile devices: A usage-driven model selection framework. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, pp 389–400

[89] Liu T, Zenke F (2020) Finding trainable sparse networks through neural tangent transfer. In: International Conference on Machine Learning, PMLR, pp 6336–6347

[90] Liu W, Anguelov D, Erhan D, et al (2016) Ssd: Single shot multibox detector. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands,

October 11–14, 2016, Proceedings, Part I 14, Springer, pp 21–37

[91] Liu X, Wu L, Dai C, et al (2021) Compressing cnns using multilevel filter pruning for the edge nodes of multimedia internet of things. IEEE Internet of Things Journal 8(14):11,041–11,051

[92] Liu Y, Guo Y, Guo J, et al (2021) Conditional automated channel pruning for deep neural networks. IEEE Signal Processing Letters 28:1275–1279

[93] Liu Y, Fan K, Wu D, et al (2023) Filter pruning by quantifying feature similarity and entropy of feature maps. Neurocomputing p 126297

[94] Liu Y, Wu D, Zhou W, et al (2023) Eacp: An effective automatic channel pruning for neural networks. Neurocomputing

[95] Liu Z, Li J, Shen Z, et al (2017) Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE international conference on computer vision, pp 2736–2744

[96] Liu Z, Sun M, Zhou T, et al (2018) Rethinking the value of network pruning. arXiv preprint arXiv:181005270

[97] Liu Z, Mu H, Zhang X, et al (2019) Metapruning: Meta learning for automatic neural network channel pruning. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 3296–3305

[98] Louati H, Louati A, Bechikh S, et al (2024) Joint filter and channel pruning of convolutional neural networks as a bi-level optimization problem. Memetic Computing pp 1–20

[99] Luo JH, Wu J (2017) An entropy-based pruning method for cnn compression. arXiv preprint arXiv:170605791

[100] Luo JH, Wu J (2020) Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. Pattern Recognition 107:107,461

[101] Luo JH, Wu J, Lin W (2017) Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision, pp 5058–5066

[102] Lym S, Choukse E, Zangeneh S, et al (2019) Prunetrain: fast neural network training by dynamic sparse model reconfiguration. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp 1–13

[103] Ma R, Miao J, Niu L, et al (2019) Transformed l1 regularization for learning sparse deep neural networks. Neural Networks 119:286–298

[104] Ma X, Li G, Liu L, et al (2022) Accelerating deep neural network filter pruning with mask-aware convolutional computations on modern cpus. Neurocomputing 505:375–387

[105] Malach E, Yehudai G, Shalev-Schwartz S, et al (2020) Proving the lottery ticket hypothesis: Pruning is all you need. In: International Conference on Machine Learning, PMLR, pp 6682–6691

[106] Manessi F, Rozza A, Bianco S, et al (2018) Automated pruning for deep neural network compression. In: 2018 24th International conference on pattern recognition (ICPR), IEEE, pp 657–664

[107] Mantena G, Sim KC (2016) Entropy-based pruning of hidden units to reduce dnn parameters. In: 2016 IEEE Spoken Language Technology Workshop (SLT), IEEE, pp 672–679

[108] Micikevicius P, Narang S, Alben J, et al (2017) Mixed precision training. arXiv preprint arXiv:171003740

[109] Mitsuno K, Kurita T (2021) Filter pruning using hierarchical group sparse regularization for deep convolutional neural

networks. In: 2020 25th international conference on pattern recognition (ICPR), IEEE, pp 1089–1095

[110] Netzer Y, Wang T, Coates A, et al (2011) Reading digits in natural images with unsupervised feature learning. " "

[111] Passos LA, Papa JP, Del Ser J, et al (2023) Multimodal audio-visual information fusion using canonical-correlated graph neural network for energy-efficient speech enhancement. Information Fusion 90:1–11

[112] Pattanayak S, Nag S, Mittal S (2021) Curating: A multi-objective based pruning technique for cnns. Journal of Systems Architecture 116:102,031

[113] Poyatos J, Molina D, Martinez AD, et al (2023) Evoprunedeeptl: An evolutionary pruning model for transfer learning based deep neural networks. Neural Networks 158:59–82

[114] Quattoni A, Torralba A (2009) Recognizing indoor scenes. In: 2009 IEEE conference on computer vision and pattern recognition, IEEE, pp 413–420

[115] Ramakrishnan RK, Sari E, Nia VP (2020) Differentiable mask for pruning convolutional and recurrent networks. In: 2020 17th Conference on Computer and Robot Vision (CRV), IEEE, pp 222–229

[116] Real E, Moore S, Selle A, et al (2017) Large-scale evolution of image classifiers. In: International Conference on Machine Learning, PMLR, pp 2902–2911

[117] Real E, Aggarwal A, Huang Y, et al (2019) Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence, pp 4780–4789

[118] Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271

[119] Redmon J, Divvala S, Girshick R, et al (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788

[120] Reed S, Zolna K, Parisotto E, et al (2022) A generalist agent. arXiv preprint arXiv:220506175

[121] Ren S, He K, Girshick R, et al (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems 28

[122] Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, Springer, pp 234–241

[123] Roy S, Panda P, Srinivasan G, et al (2020) Pruning filters while training for efficiently optimizing deep learning networks. In: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–7

[124] Russakovsky O, Deng J, Su H, et al (2015) Imagenet large scale visual recognition challenge. International journal of computer vision 115:211–252

[125] Sarvani C, Ghorai M, Dubey SR, et al (2022) Hrel: Filter pruning based on high relevance between activation maps and class labels. Neural Networks 147:186–197

[126] Schwartz R, Dodge J, Smith NA, et al (2020) Green ai. Communications of the ACM 63(12):54–63

[127] Shang H, Wu JL, Hong W, et al (2022) Neural network pruning by cooperative coevolution. arXiv preprint arXiv:220405639

[128] Shao M, Dai J, Wang R, et al (2022) Cshe: network pruning by using cluster similarity and matrix eigenvalues. International Journal of Machine Learning and Cybernetics pp

1–12

[129] Shi C, Hao Y, Li G, et al (2023) Vngep: Filter pruning based on von neumann graph entropy. Neurocomputing

[130] Shu H, Wang Y, Jia X, et al (2019) Co-evolutionary compression for unpaired image translation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 3235–3244

[131] Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. ICLR, pp 1-14

[132] Singh P, Verma VK, Rai P, et al (2020) Leveraging filter correlations for deep model compression. In: Proceedings of the IEEE/CVF Winter Conference on applications of computer vision, pp 835–844

[133] Skandha SS, Agarwal M, Utkarsh K, et al (2022) A novel genetic algorithm-based approach for compression and acceleration of deep learning convolution neural network: an application in computer tomography lung cancer data. Neural Computing and Applications 34(23):20,915–20,937

[134] Strubell E, Ganesh A, McCallum A (2019) Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:190602243

[135] Sun X, Ren X, Ma S, et al (2017) meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. In: International Conference on Machine Learning, PMLR, pp 3299–3308

[136] Szegedy C, Liu W, Jia Y, et al (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9

[137] Tan M, Le Q (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning, PMLR, pp 6105–6114

[138] Tan M, Chen B, Pang R, et al (2019) Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2820–2828

[139] Tang Z, Luo L, Xie B, et al (2022) Automatic sparse connectivity learning for neural networks. IEEE Transactions on Neural Networks and Learning Systems

[140] Tian G, Chen J, Zeng X, et al (2021) Pruning by training: a novel deep neural network compression framework for image processing. IEEE Signal Processing Letters 28:344–348

[141] Tmamna J, Ayed EB, Ayed MB (2021) Neural network pruning based on improved constrained particle swarm optimization. In: Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part VI 28, Springer, pp 315–322

[142] Tmamna J, Ayed EB, Fourati R, et al (2023) An automatic vision transformer pruning method based on binary particle swarm optimization. In: 2023 IEEE Symposium on Computers and Communications (ISCC), IEEE, pp 727–732

[143] Vadera S, Ameen S (2022) Methods for pruning deep neural networks. IEEE Access 10:63,280–63,300

[144] Verma VK, Singh P, Namboodri V, et al (2020) A" network pruning network"approach to deep model compression. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp 3009–3018

[145] Wah C, Branson S, Welinder P, et al (2011) The caltech-ucsd birds-200-2011 dataset. " "

[146] Wang C, Zhang G, Grosse R (2020) Picking winning tickets before training by preserving gradient flow. arXiv preprint arXiv:200207376

[147] Wang H, Qin C, Bai Y, et al (2022) Recent advances on neural network pruning at initialization. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, Vienna, Austria, pp 23–29

[148] Wang W, Zhao S, Chen M, et al (2019) Dbp: Discrimination based block-level pruning for deep model acceleration. arXiv preprint arXiv:191210178

[149] Wang Z, Liu X, Huang L, et al (2021) Model pruning based on quantified similarity of feature maps. arXiv preprint arXiv:210506052

[150] Wen W, Wu C, Wang Y, et al (2016) Learning structured sparsity in deep neural networks. Advances in neural information processing systems 29

[151] Wimmer P, Mehnert J, Condurache AP (2023) Dimensionality reduced training by pruning and freezing parts of a deep neural network: a survey. Artificial Intelligence Review pp 1–39

[152] Xiao X, Wang Z, Rajasekaran S (2019) Autoprune: Automatic network pruning by regularizing auxiliary parameters. Advances in neural information processing systems 32

[153] Xie X, Zhang H, Wang J, et al (2019) Learning optimized structure of neural networks by hidden node pruning with $l\_\{1\}$ regularization. IEEE Transactions on cybernetics 50(3):1333–1346

[154] Xu X, Chen J, Li Z, et al (2024) Towards efficient filter pruning via adaptive automatic structure search. Engineering Applications of Artificial Intelligence 133:108,398

[155] Yang C, Liu H (2022) Channel pruning based on convolutional neural network sensitivity. Neurocomputing 507:97–106

[156] Yang C, Yang Z, Khattak AM, et al (2019) Structured pruning of convolutional neural networks via l1 regularization. IEEE Access 7:106,385–106,394

[157] Yang L, Gu S, Shen C, et al (2024) Soft independence guided filter pruning. Pattern Recognition p 110488

[158] Yang W, Xiao Y (2022) Structured pruning via feature channels similarity and mutual learning for convolutional neural network compression. Applied Intelligence 52(12):14,560–14,570

[159] Yang W, Jin L, Wang S, et al (2019) Thinning of convolutional neural network with mixed pruning. IET Image Processing 13(5):779–784

[160] Yeom SK, Seegerer P, Lapuschkin S, et al (2021) Pruning by explaining: A novel criterion for deep neural network pruning. Pattern Recognition 115:107,899

[161] Yim J, Joo D, Bae J, et al (2017) A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4133–4141

[162] Yim J, Joo D, Bae J, et al (2017) A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4133–4141

[163] Yin H, Molchanov P, Alvarez JM, et al (2020) Dreaming to distill: Data-free knowledge transfer via deepinversion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 8715–8724

[164] You Z, Yan K, Ye J, et al (2019) Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. Advances in neural information processing systems 32

[165] Yu L, Xiang W (2023) X-pruner: explainable pruning for vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 24,355–24,363

[166] Yue L, Weibin Z, Lin S (2019) Really should we pruning after model be totally trained? pruning based on a small amount of training. arXiv preprint arXiv:190108455

[167] Zhang P, Tian C, Zhao L, et al (2024) A multi-granularity cnn pruning framework via deformable soft mask with joint training. Neurocomputing 572:127,189

[168] Zhang S, Gao M, Ni Q, et al (2023) Filter pruning with uniqueness mechanism in the frequency domain for efficient neural networks. Neurocomputing 530:116–124

[169] Zhang W, Wang Z (2022) Fpfs: Filter-level pruning via distance weight measuring filter similarity. Neurocomputing 512:40–51

[170] Zhang Y, Freris NM (2023) Adaptive filter pruning via sensitivity feedback. IEEE Transactions on Neural Networks and Learning Systems

[171] Zhang Y, Zhen Y, He Z, et al (2021) Improvement of efficiency in evolutionary pruning. In: 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–8

[172] Zhang Z, Li Z, Lin L, et al (2020) Metaselection: metaheuristic sub-structure selection for neural network pruning using evolutionary algorithm. In: ECAI 2020. IOS Press, p 2808–2815

[173] Zhao H, Sun X, Dong J, et al (2020) Highlight every step: Knowledge distillation via collaborative teaching. IEEE Transactions on Cybernetics 52(4):2070–2081

[174] Zhou Q, Huang Z, Ding M, et al (2023) Medical image classification using light-weight cnn with spiking cortical model based attention module. IEEE Journal of Biomedical and Health Informatics 27(4):1991–2002

[175] Zhou Y, Yen GG, Yi Z (2019) Evolutionary compression of deep neural networks for biomedical image segmentation. IEEE transactions on neural networks and learning systems 31(8):2916–2929

[176] Zhou Y, Yen GG, Yi Z (2019) A knee-guided evolutionary algorithm for compressing deep neural networks. IEEE transactions on cybernetics 51(3):1626–1638

[177] Zhou Y, Yen GG, Yi Z (2021) Evolutionary shallowing deep neural networks at block levels. IEEE Transactions on Neural Networks and Learning Systems 33(9):4635–4647

[178] Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. arXiv preprint arXiv:161101578

[179] Zoph B, Vasudevan V, Shlens J, et al (2018) Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8697–8710