# Model Compression Techniques in Deep Neural Networks*

Mubarek Mohammed Yesuf[1][0000−0002−0145−1810] and Beakal Gizachew Assefa[2][0000−0003−3065−3881]

[1] Information Network Security Agency and Addis Ababa Institute of Technology, AAU, Addis Ababa, Ethiopia
mubarek.mohammed@aait.edu.et
[2] Addis Ababa Institute of Technology, AAU, Addis Ababa, Ethiopia
beakal.gizachew@aait.edu.et

**Abstract.** With the current set-up, the success of Deep Neural Network models is highly tied to their size. Although this property might help them improve their performance, it makes them difficult to train, deploy them on resource-constrained machines, and iterate on experiments. There is also a growing concern about their environmental and economic impacts. Model compression is a set of techniques that are applied to reduce the size of models without a significant loss in performance. This survey presents state-of-the-art model compression methods. The paper also puts forward open research problems for further study. The paper also compares and contrasts the relationship of model compression with other concepts beyond size reduction. It also entails a separate section for hybrid methods that combine compression methods.

**Keywords:** Model compression · Network pruning · Knowledge Distillation · Quantization · Neural Networks · Deep learning · Artificial Intelligence

## 1 Introduction

Breakthrough advances in AI algorithms are consequences of Neural Networks which are the foundations for Deep Learning (DL) which is a family of Machine Learning algorithms behind successful Artificial Intelligence (AI) innovation in tasks like voice recognition,image classification[41], human language understanding [16], etc. Deep Learning algorithms are based on the universal approximation theorem[33], which guarantees that Neural Networks, grouped in some way, can compute any continuous function, and the hierarchical representation of information inspired from the human brain[5]. These ideas led to incentives the over-parameterization of networks to ensure performance because the exact number of parameters a specific Deep Learning (DL) model needs for a certain problem is still a hyper-parameter, a choice of the designer. Therefore, in-order to

* Supported by Ethiopian Artificial Intelligence Institute.

in crease a model's representation capacity and thus performance, the design of models is growing larger and larger as the tasks they are being applied to grows in complexity [6]. How complex a model is can be measured in a number of ways. The common measures are the total number of of learnable parameters, useful to measure the memory footprint or size of the model, and floating point operations per second(FLOPs), useful in fields of computations that require floating-point calculations.

Most bench mark models and datasets are related to computer vision, a subfields of AI that tries to mimic the human vision. A CNN is a type of deep learning architecture designed for spatial data such as an image. Prominent examples are, Alexnet [41] and VGG16 [64], which are variants of CNNs, are winners of Imagenet[41],a state of the art computer vision dataset with more than 14 million well labeled images intended to serve as a benchmark in computer vision applications. Figure 6 below shows the evolution of these benchmark models and their computational burdens expressed in Giga FLoating-point OPerations (GFLOPs) along with their Top-1(left) and Top-2 accuracy.

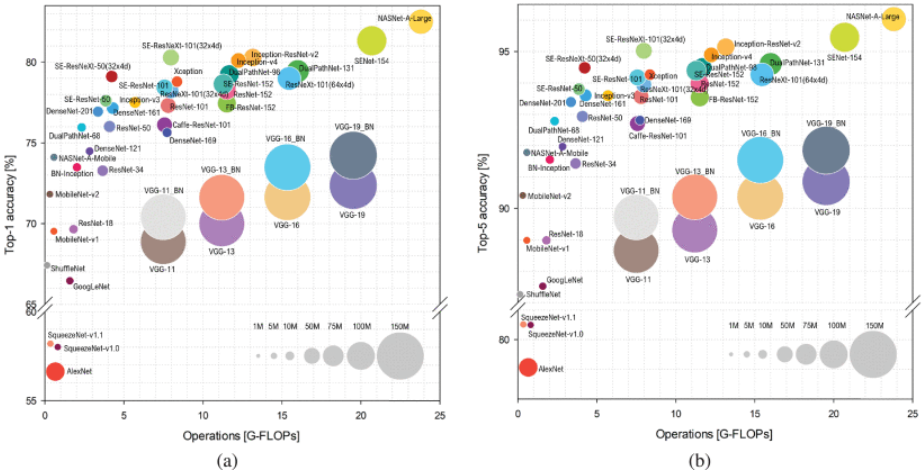


**Fig. 1.** Evolution of size of models [6]

However, there is a tradeoff: these large models are performant but they come at a cost. Challenges of such big models include longer training, inference and iteration time, difficulty to deploy on resource constrained and edge machines, and economic and environmental impact [56]. This raises the question if there is a way to work around the trade-off. This is where Model Compression comes in, it is the idea of making models small without losing performance. This paper presents a comprehensive survey of deep-learning model compression.

Model Compression is a set of techniques [9] for reducing the size of large models without a significant loss in performance. Its necessity is increasing in

parallel with the advancement and complexity of large models. Even though there recent attempts to train a smaller networks from the begging [19], mostly, Model Compression is applied after training a bigger model as the model needs much less parameters for inference than for learning.

There are numerous Model Compression methods in literature but for the purpose of this paper, they can be classified in four major parts: Pruning, Knowledge Distillation (KD), Quantization, and other methods. Pruning is removing an unwanted structure from a trained network. The way to determine the what part of the network is unwanted,how to and when to remove create different variants of Pruning. Knowledge Distillation (KD) is a mechanism transferring the knowledge of a bigger network onto a smaller network. Quantization is reducing the number of bits model parts require to be represented. It is similar to approximation. The rest of Model Compression methods other than Pruning, Knowledge Distillation (KD) and Quantization are, for the purpose of this paper, fused in one section and are referred to as Other methods. Weight decomposition methods deserve to be to have their own section, but this arrangement makes the paper simpler for the reader.

The contributions of the paper are as follows

- Conduct a comprehensive review of deep learning model compression methods.
- To the best of our knowledge, we are the first to categorize, systematically compare, and discuss the pros and cons of each method along with their consequences beyond size reduction and their combinations.
- Discuss open research problems for further study on the topic of deep neural network model compression.

The remainder of the article is organized as follows. The next section, section two, reviews prominent works in Model Compression literature by classifying them into four major areas as Pruning, Knowledge Distillation, Quantization and Other methods. Section three is on Hybrid Model Compression techniques which is about methods that combine existing ones for a better outcome. The effect of Model Compression beyond size reduction is discussed in section four. Discussion, open research problems, and conclusion are presented in the last three sections, section five, six and seven respectively.

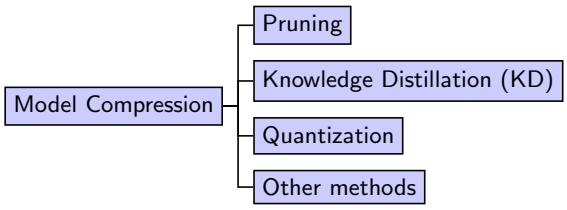


**Fig. 2.** The taxonomy and categories of machine learning model compression methods

Figure 2 shows the categorization of modle

## 2 Model Compression Methods

### 2.1 Pruning

The oldest, most studied and intuitive method of reducing model size is pruning. It is literally removing a weight or a node from a network based on how important it is. The vanilla pruning method has three main stages: initializing, training, pruning and retraining(fine tuning). How to prune, what to prune, and other developments on the conventional pipeline categorize the pruning methods in literature.

#### 2.1.1 Early works

The earliest methods were of two types mainly: Saliency based where node or weight is checked for importance, and penalty based, where the a penalty term is added on the loss function that will create weight decay and thus create sparsity[61]. Saliency metric based methods check each weight or node whether it is important or not. The metric is at the heart of saliency based methods and these metrics have evolved to be more surgical with developments in algorithms. Magnitude based pruning is the simply removing the smallest link(weight) in each iteration. A better attempt to get out of the 'magnitude is importance' paradigm is Optimal Brain Damage (OBD) [42] where saliency is defined as the change of the error function when pruning a certain parameter by estimating the second derivative of the error with respect to the weight which is improved in Optimal Brain Surgeon [28] where the exact computation of the Hesse-Matrix is introduced for a surgical weight removal at a cost of computation. Skeletonization [48] prunes units (neurons), including input features, by approximating the change of the objective function when the unit is removed. Other methods use statistical methods to identify non-contributing Units, Weights that don't vary their output, always show the same output as another unit of the same layer or always show the opposite output of another unit of the same layer are the target of this method. Currently, there are over three hundred saliency metrics in literature [57] that evolved out of the earliest methods with advances in neural network algorithms.

#### 2.1.2 Explainability Based Methods

A more 'educated' way of dubbing a parameter as unimportant based on interpretability or explainability. Explainability is the notion of trying to explain the 'decision' of a model. This is equivalent to asking the question of what part of the input cause a specific prediction. Currently, that is a hot topic as applications of Artificial Intelligence (AI) prevail in more and more sectors including high stake areas. There are well established methods out there and a couple of them have been used in the context of pruning. An area of growing interest is

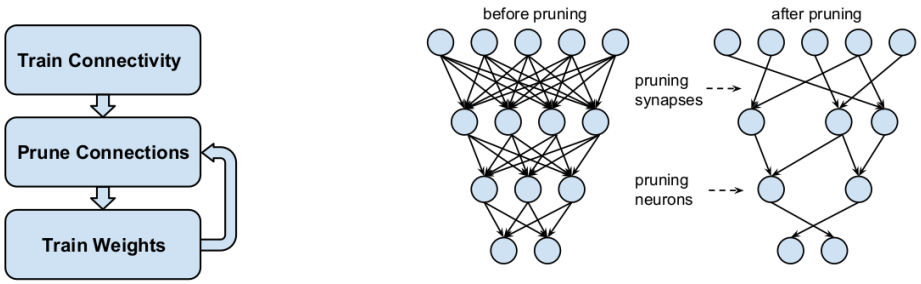


**Fig. 3.** Iterative Pruning (left) and Pruning(right) [27]

what is being popularized as Interpretable Pruning, an attempt to know what is actually being pruned. Layer-wise Relevance Propagation (LRP) algorithm is proposed in [76] to assign relevance scores to each and every unit in the network. The work in [78] demonstrated a Neuron Importance Score Propagation (NISP) algorithm to propagate the importance scores of final responses to every neuron in the network and then prune the CNN filter withe the least importance. Interpretability based Filter Pruning (IFP) is introduced in [75] that utilizes Activation Maximization, a form of feature visualization method after each activation layer of a CNN, to identify non essential filters. The method identified filters that learn redundant features and pruned them. An attempt to dynamically determine what neurons to skip at inference time is demonstrated by [70]. This method incorporated a decision unit in the before every CNN layer which take the input and give output an indication of the most probable channel. In convolutional neural networks, additional to nodes and weights, now there are channel pruning methods ( [31], [43] ) and filter pruning [29] pruning methods. Such methods came to be known as structured pruning, removing structures, as opposed to unstructured pruning, including removing weights or nodes as in the legacy approaches.

### 2.1.3 Modern Prunig Based

Modern pruning-based compression approaches vary not just in the Saliency metrics they use or other approach, but also in the overall pipeline of pruning. The 'lottery ticket hypothesis' [19], whose stronger version was supported in [46], states that random initialization of a network contains a small subnetwork (" the winning tickets") that, when trained in isolation, can compete with the performance of the original network. A rather bold finding in by [59] states that there is a randomly initialized subnetwork that even without training can perform as good as the original. The recent work [71], termed pruning from scratch, questions the need for training the network and proposes a different pipeline as: random initialization, pruning and then training. The work in [44], raises the question why train a big model to convergence while it is possible to attain better performance by training it for free iterations and prune it.

A new perspective in pruning is treating the task as a search in the space of sub-networks. This is a consequence of advances in Neural Architecture Search, an emerging area of study in the study of neural network's. The researchers in [18],applied neural architecture search to search directly for a network with flexible channel and layer sizes where the number of the channels/layers is learned by minimizing the loss of the pruned networks.

Pruning methods rely on manual design to get optimal performance. An emerging approach is to automate the process of pruning using Reinforcement Learning methods following advances in the area. RL based pruning methods formulate the problem of removing ad node or a weight as a Markov Decision Process. These methods of pruning differ in their design of the RL system which essentially requires to define a state space, an action space, and a reward. The action and state space are obvious enough as they are almost always the set of all 'removing' and the set of all possible sub networks respectively. The rewards for pruning can be a dense [25], appearing after every action is taken, or sparse[30], coming or appearing to the agent after a certain sequence of actions. In [25] a magnitude is determined to in each step to remove or leave a layer. Almost all RL based methods incorporate structured pruning for the actual pruning, which can be saliency or magnitude based. This is because RL methods are by nature iteration intensive and if individual parameters are considered for each action-state pair, the complexity will be exponential.

#### 2.1.4 Findings in Pruning

The most important gap common in all the pruning methods is that it is hard to compare different pruning variants due to lack of benchmark metric. The only benchmark effort is Shrinkbench [7], a framework aiming to provide pruning performance comparisons.

There is a critical difference between unstructured and structured pruning. Unstructured pruning, is implemented by making individual parameters to be pruned zero. This makes unstructured pruning almost surgical and more accurate in identifying what to remove. But this implementation doesn't actually reduce the computation time since the GPU will do the operation anyway. On the other hand, structured pruning will remove a complete layer or a channel or a filter. And then the rest of the connections will be connected afterwards. This increases efficiency both pruning time and implementation time. But this too has its own consequences as it is not surgical. The layer or channel to be removed might still have important weights that are needed for inference.

Another common limitation of pruning methods, which can be caused by the lack of a common benchmark, is that they are architecture dependent. Almost all of them involve at least some amount of fine-tuning or retraining which requires having original data to train network is trained on which in some cases might not be available.

Considering randomly dropping weights or magnitude based pruning, it is justified to say that pruning is the fastest compression method.

### 2.2 Knowledge Distillation

Knowledge distillation is a relatively new compression technique originally introduced in [9] and reinvented in [32]. In [9], they showed that it is possible to compress the knowledge in an ensemble into a single model. The main idea is that there will be a bigger model, which is assumed to be performing very well, providing the labels, for a smaller network. For each observation x and its label y out of the dataset the original network was trained on, the smaller network will use the prediction or inference of the bigger network as a label for x instead of y. In the scheme, the bigger model provides labels for the smaller network to be trained with. Soft labels rather than hard labels, with many zeros and a single one to identify the class. The soft labels, which are essentially the mistakes of the good performing model, are further termed as Dark Knowledge.


**Fig. 4.** Soft labels [17]

The intuition is an image of a dog has a larger chance of being mistaken as a cat than as a car. These probabilities of the mistake class actually entail knowledge of the bigger model but are usually too small. For this, the knowledge distillation method involves the use of a constant value named Temperature, T, which magnifies these false probabilities. This temperature is used to magnify the dark knowledge to be transferred. Each output in training, with logit z for the i-th class, and a temperature T, is computed as :

$$p(z_i, T) = \frac{exp(z_i/T)}{\sum exp(z_j/T)}$$

( Knowledge distillation is flexible and model agnostic, any model can be used as teacher or a student. In literature, the bigger model is termed as the teacher, the small network is termed as the student network and the dataset used is the transfer set.

Ever since the original method [32], a wave of different variants of knowledge distillation were proposed that differ in : in the type of knowledge that is to be transferred, the scheme, the type of student-teacher architecture, the purpose of the knowledge distillation used, and the implementation setting [24].

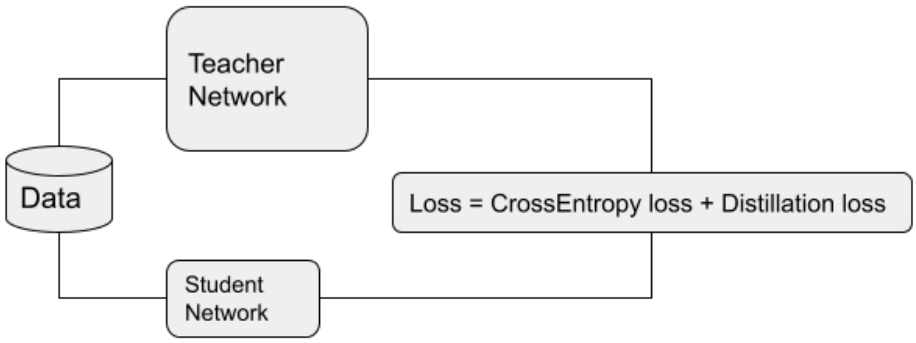


**Fig. 5.** Training in Knowledge Distillation (KD)

A key part of the Knowledge Distillation mechanism, as shown in the figure 5, is loss part which is made up of Crossentropy loss and a distillation loss. The distillation loss is the Kullback-Leibler (KL) divergence loss of the student predictions and the teacher predictions that are both probability distributions.

A powerful adaptation of knowledge distillation, Fitnets [62], trains a student, deeper and thinner than the teacher which is unlike [32], using not only the outputs but also the intermediate representations, features, learned by the teacher as hints to improve the training process and final performance of the student. This work opened a whole new sub-catagory of knowledge distillation termed as feature distillation making the original to be called response based distillation [24]. Response-based knowledge distillation, is also limited to the supervised learning as logits,the soft labels, are probability distributions.

Relationship distillation extends feature distillation by using outputs of specific layers in the teacher model and exploring the relationships between different layers or data samples [55].

Knowledge distillation is a very flexible technique that is being stretched in many dimensions.The student network size is not limited. The work in [21] showed than the student can have the same size as the teacher and outperform the teacher. An assistant network, a size that is less than the teacher and smaller than the student, can also be introduced in the architecture to help the small student network catch up with the big teacher network [47]. There are also variants based on the loss function used [67]. The relatively recent variant of knowledge distillation presented in [49] demonstrated the capability of the method even when the number of classes is too small and the Dark Knowledge, the transferable information from the teacher, which is the function of the number of classes, is very limited.

There are even attempts to do knowledge distillation without the existence of the data, the transfer set [77]. The general approach followed by most of data free methods is first to generate, usually by what is commonly known

as Generative Adversarial Network, some-sort of pseudo-data or synthetic data from the teacher model and then train the student with it [77] and [72].

The most recent trend in knowledge distillation is what is known as Zero shot knowledge distillation [50] where the only available information is the model itself. This is almost orthogonal to the previous data free techniques as they need at least meta data or other information to start with. In case of [50], the student network is trained with a synthetic data generated from what are called class impressions extracted from the teacher network.

**2.2.1 Findings in Knowledge Distillation** Knowledge distillation methods actually construct a brand new smaller model that tries to mimic the bigger model. This means training a new neural network from scratch which will take as much time as the training time for the teacher. The loss have to be a softmax loss function whose output is a probability distribution. Knowledge distillation methods, for compressing a neural network, doesn't give a bigger size reduction. Some might even need a teacher assistant to catch up to the teacher[47].

In Knowledge Distillation (KD), the added term, the distillation loss which is seen above5, does serve as a regularizer [63], the exact part of dark knowledge the student is taking advantage of is still being studied[79].

### 2.3 Quantization

Quantization is a fancier form of rounding up a number in the parameter. But in Neural Network setting, quantization is about [23] rounding up the parameters to compresses the original network by reducing the number of bits required to represent each weight, which is usually in 32 bits floating point representations, to limited precision, commonly in 8 bits or lower representations. It is more of an acceleration technique than actual size reduction. It's foundation is rooted in how a human brain stores and encodes information in a discrete way[22]. The basic method of quantization is to train a model under normal setup and then to quantize each parameter, activation or even input, using some sort of function or mapping, which is sometimes referred to as a quantization operator [22]. Normally, what is known as the clipping range, a range to make sure the individual weights will lie in, is determined before hand. The common formula looks something like :

$$Q(r) = Int(\frac{r}{S}) - Z[22]$$

where Q is the quantization operator, r is a real valued scheme, since the clipping range is not necessarily input (activation or weight), S is a real valued scaling factor, and Z is an integer.

In literature, the variants of quantization differ according to which parameters to quantize, how to quantize, whether to do it in training time or after training time, and other factors which correspond to the variables S, Z in the quantization equation. S itself is determined by the boundary values of the clipping range.

Depending on whether the clipping range is calculated for each dynamically or statically, quantization can be dynamic or static. The work on [36] made sure only 8-bit integers are used in the network at inference time.

When fine tuning is needed after quantization, Quantization Aware Training is used where a re-training is done with floating point propagation but the weight are quantized back after each update. Post Training Quantization is quantization without retraining [4], [10]. It is preferred when QAT is too complex to implement.

The ideal Quantization is, Binarization, making the parameters of a network binary as they are introduced in [14]. Its advantage is two folds: It can be represented in 1 bit and it makes computation a lot smoother. In fact, the most appealing feature of this method, given it works properly, is that it might even completely get rid of the need of multiplication at inference time as in [15]. This is an ideal situation as neural networks are by design multiplication intensive. This is possible by replacing the dot product in conventional neural networks with bitwise operators [14]. The work in [14] demonstrates a method to train networks with binary weights. In [60], the researchers showed CNNs with only binary weights and demonstrated a binary quantized version of Alexnet[41], with 32 times smaller, with comparable accuracy as the original version. They further demonstrated a constraints not only the weights, but the inputs themselves to be binary would give comparable accuracy with 58 times inference efficient. Beyond binarization, an alternative network is a ternary network, where the weights f the network are limited to three values instead of two as in binary. Neither binary nor ternary Quantization methods are trainable with back-propagation with gradient descent.

Recent, advances introduce Differentiable Quantization in order to learn the hyperparameters of Quantization rather [81].

**2.3.1 Findings in Quantization** Quantization methods make weights discrete. They might also damage the precision of important weights. There are ways to bypass that, dynamic, Asymmetric Quantization or Quantization Aware Training), but they come at the expense of computation which is what compression was needed for in the first place.

### 2.4 Other Compression Methods

In general, the goal of compression is intuitive: reduce size. In reduce size with accuracy. Thus, people have tried whatever they think would work to achieve that goal. Listed in this section are methods that are both common, Tensor Decomposition, and rare, like Information theoretic.

Tensor decomposition is a popular method to reduce a model size. A tensor is just a matrix in higher dimension. Tensor decomposition is a scaled up version of the common matrix decomposition in linear algebra. Matrix decomposition methods like Singular Value Decomposition and Principal component analysis, have their analogous versions. To make compression, these methods take the

weights of a model to form a Tensor. Tensor decomposition, the core of these methods, is an existing approach in other fields and common decomposition methods Canonical Polyadic, Tensor Train, and the Tucker decomposition are applied to compress models. The works in [51], [11] and [40] are iconic examples of these methods. These are almost always applied in Multi Layer Perceptrons or () layers of CNNs. Large scale matrix manipulation involved is their disadvantages. They are also applied on specific architectures.

Weight sharing approaches are somehow ways of making Convolutional Neural Network (CNN) variants efficient. The works [66], Inception [80],MobileNet [34] and SqueezNet [35] are examples of these methods. One of the most influential work [35] achieved Alexnet[41] level accuracy with 50 times fewer parameters on the ImageNet dataset [41]. These family of compression methods are more of careful design choices that turned out to be efficient.

An unique method is presented in [73] used the then state of the art lossless video encoding and compression technique to compress a model. It is an information theoretic approach that formulated the model compression problem as an information source coding one. It used Context-Adaptive Binary Arithmetic Coding (CABAC) a video coding standard H.264/AVC to encode and the quantize the weights of the neural network but the encoding takes long time.

## 3 Hybrid Model Compression Methods

Fortunately, Model Compression methods are not disjoint: they can be combined. The resulting method would be more complex than the individual ones but in cases where that is not an issue, combining hybrid methods is beneficial. In fact, in certain cases like [1], they can achieve a symbiotic relationship where the problems of one are addressed by the other and vice-versa [1]. There are a number of works that try to combine one or more of them and this section will present prominent ones.

### 3.1 Pruning and Quantization

Pruning and Quantization the most related ones especially unstructured Pruning where it is simply setting a parameter zero. The most prominent work not just in hybrid paradigm but also in the area of Model Compression is Deep Compression framework [26] where iterative Pruning, Qauntization and Encoding are applied one after the other. Its Pruning part is the same as [27].

The intuitive relationship between Pruning and Quantization is being backed by a recent advances in Quantization that study differentiable hyperparamters in Quantization as a means to unify them [69],[68].

### 3.2 Pruning and Knowledge Distillation

There is also a positive relationship between Pruning and Knowledge Distillation and it has been recognized relatively early in [39] where Pruning is applied on

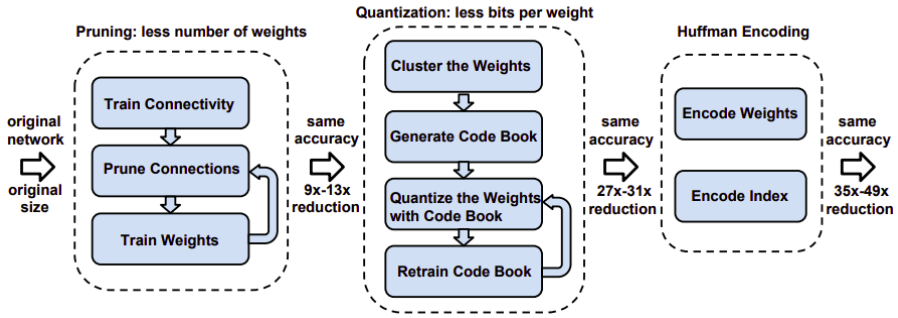


**Fig. 6.** Deep Compression framework [26]

distilled student networks on the task of Neural Machine Translation. Recent works are more generic than applied. A work demonstrated in [54] showed, a student can learn better from pruned teacher which serves as a better regularizer than otherwise in the context of Knowledge Distillation. The converse of it has also been shown in [12] where the authors argue that the performance of a pruned network is better restored when the student is fine-tuned with Knowledge Distillation (KD) than vanilla training. A good example of a symbiotic relationship between the two is [1] where unprunable parameters are compressed with Knowledge Distillation (KD) and potentially redundancy is addressed by Pruning[1].

### 3.3 Quantization and Knowledge Distillation

An early major work in the combination of Quantization and Knowledge Distillation is the application of Quantization for the choice of a student model for Knowledge Distillation [58] where a the knowledge of a Quantized teacher is transferred to a Quantized student. In this setting, Knowledge Distillation helps Quantization restore better performance. Conversely, Quantization also benefits Knowledge Distillation (KD) as in [8] where it has been used to enable on device Knowledge Distillation by introducing noise that mimic the existence of a bigger teacher model. This addresses the problem of missing teacher on edge devices to do vanilla Knowledge Distillation (KD) and might even put the need of the teacher in the first place in question. Their bond is strengthened by Quantization Aware Knowledge Distillation [38] that aims to solve the problem of performance degradation in Quantized distillation [58].

## 4 Model Compression Beyond Size Reduction

The purpose of compressing a model is to reduce its size. But the size reduction can have both positive and negative consequences on other aspects of a network

**Table 1.** Summary of Hybrid Model Compression

| **Hybrid Compression** | **Remark** |
|---|---|
| Pruning and Quantization | They are applied together almost everywhere [26] and are so similar there are work that attempt to unify them [69],[68] |
| Pruning and Knowledge Distillation (KD) | Mutually beneficial result. One solves the challenges of the other.[1], [54] ,[39] |
| Quantization and Knowledge Distillation | Distillation restores performance damaged by Quantization [58]. Quantization helps mimic a non existent teacher [8] |

like explainability. In the case where it benefits, it will be an extra advantage. In fact, in some cases, like [42], the compression can even be done for the by-product. But in the cases where it is disadvantageous, as in on explainability[37], there is a tradeoff to be addressed appropriately. Basically, all decisions that were made about the network before compressing it can be questioned after the compression, but mentioned here are four of them for there is a lack of enough work in the area.

### 4.1 Reducing Overfiting

Ideally a model is expected to generalize and not overfit. Extreme pruning does damage generalization[27] but a certain level of pruning also helps parameters learn representations independently and serve as a means to introduce noise in the network which serves as a regularizer which in turn reduce overfiting. Overfiting is when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance. The most famous applications of compression, especially pruning, is generalization. In fact, in early days neural networks, the purpose of pruning was to reduce overfitting [42]. Recently, dropouts, randomly dropping out weights from the network with a certain probability, [41] enabled for models to go deeper than usual and reignited the consideration of compression as a cure for overfiting. Dramatic size reduction with Knowledge Distillation (KD)[62] impacts generalization negatively.

### 4.2 Explainability

Explainability or interpretability is an effort to try to understand the decisions of neural networks. This issue is getting more and more attention due to high stake applications of neural networks. Compression is also tied with explainability. In [20], the researchers distilled the knowledge of a high performing neural network into a decision tree model which is explainable inherently. A later work [45] generalized the application of knowledge distillation for the purposed of interpertability by formulating the problem as a multi-output regression problem. The result is a Decision tree that performs better than one trained on the data directly but not better than the original neural network. Thus, trading a little

bit accuracy for interpretability. But this is a double-edged phenomenon because compressing a trained model impacts explanations as it can be seen [37] and [53] where the authors suggested explanation preserving methods. There is a lack of detailed work between compression and explainability. For example, what model compression does to Mechanstic Interpretability, an effort trying to understand what happens inside Neural Networks, remained a mystery[52].

### 4.3 Neural Architecture Search

Neural Architecture Search is a relatively recent approach in the Artificial Intelligence (AI) community. It is an attempt to find a an optimal architecture in an educated way. This is because existing novel architectures are almost human choices and could have been different. The relationship of Neural Architecture Search with model compression is also intuitive and well recognised in literature [13]. The task of optimal compression can be seen as a search in the space of sub-architectures. For example the task pf pruning can be taken as a search in the space of architectures that are sub networks of the original network. The works in [74] and [2] directly demonstrate this concept.

### 4.4 Algorithmic Fairness

Algorithmic fairness has become increasingly important due to the increasing impact of Artificial Intelligence (AI) on our society. Particularly, as they are being adopted in various fields of high social importance or automate decision-making, the question of how fair the algorithm is critical now more than ever. Since Model Compression is becoming a defacto component of Machine Learning (ML) deployment, its impact on fairness have to be examined. A recent work aimed at this problem reports that model compression, Pruning and weight decomposition, seem to exacerbate the bias in a network[65].

**Table 2.** Summary of Model Compression and its consequences

| **Network element** | **Relationship** |
|---|---|
| Overfiting | Slight pruning is known to have positive impact [42]but extreme compression of both Pruning [27]and Knowledge Distillation (KD) [62] damage it. |
| Explainability | Both Pruning and Knowledge Distillation damage Explainability(Attribution) but Knowledge Distillation indirectly solves the problem of Explainability by transferring knowledge to an interptetable model[37],[53] |
| Neural Architecture Search | It has a two way positive relationship with Pruning and Knowledge Distillation [13],[2]. |
| Algorithmic Bias | All compression methods exacerbate the existing bias[65] |

## 5 Findings and Open Research Problems

In general, model compression can be seen in two broad categories. The methods in the first category transform a trained model into a small model using a certain technique. Pruning, Quantization and Tensor Decomposition methods can seen as model transformation methods. They start with a trained model and transom it to a optimized version of itself. The methods in the second category try to create a new but smaller model from scratch. Knowledge distillation can be seen as creating a model from scratch, in the second category. Any other design choices that can reduce model size can be categorized in the second category.

Most compression methods are vision native, they are made for computer vision applications especially Convolutional Neural Networks (CNN) based architectures. That is most likely because vision data is abundant, to make inference at edge devices where the camera resides, or to take advantage of compression methods that remove structures at a time.

Another pattern observed is that there is similarity across the techniques, advantages and disadvantages of both pruning and quantization. Conventional Pruning and Quantization are both port training. Applying them while training is possible but it has an additional cost of complexity. Particularly, Unstructured pruning and Uniform Quanitzation are analogous. In both cases, the more surgical one methods increase in complexity. But quantization has more headaches to be surgical. For this reason, they are heavily applied in structured in Pruning( in uniform in Quantization) in practical settings. They are both looking at AutoML solutions such as Reinforcement Learning (RL) and Neural Architecture search (NAS) for solution that take advantage of two of their variants.

At last, there is no size fits all compression method. One has to find the Goldilocks model compression technique for the task at hand.

**Table 3.** Summary of model compression methods

| **Method** | **Description** | **Remark** |
|---|---|---|
| Pruning | Removing parameters by setting them to zero | The oldest, fastest and one with the biggest compression rate. |
| Knowledge distillation | Train a new smaller model with the predictions the bigger model | Most versatile and model agnostic but works with only certain types of loss functions |
| Quantization | Reduce the amount of bits required to represent the parameters | Work well with pruning, effective to reduce memory size as well as computation |
| Other methods | Methods that reduce modeluu size with techniques other than the above three. | Efficient architecture designs, inspired by file compression methods, or matrix decomposition based methods |

There are numerous open issues in Model Compression that can be formulated as a valid research questions. They are presented in this section according to the classification used in the paper.

### 5.1 Knowledge Distillation (KD)

Knowledge Distillation (KD) is one of the major compression methods. In its framework as can be seen in the figure5, the added term in the loss part is the knowledge that is assumed to be being helpful for the teacher. It does serves as a regularizer [32]. But the performance and even the need of the teacher network itself, especially in response based Knowledge Distillation (KD), is being questioned repeatedly [8], [79],[63], [3]. What part of the dark knowledge the student takes advantage of is still an open problem.

There are also understudied variants of Knowledge Distillation. An example is Subclass distillation [49] that entail a powerful concept. Especially, its converse, extracting specialists from a bigger model, seems a fruitful direction. Zero shot knowledge distillation methods, as discussed before, use random inputs to the trained model to generate psudo data to train the small network with. Studying how the result will turn out for a curated input is also a possible direction.

### 5.2 Hybrid Methods

As described in section three, combining compression methods can be beneficial if the added complexity is not an issue. In general, there could combinations in two ways: combining variants of the same method to take advantage of them and combinations between different compression methods as in [26]. Thus,a study on the combination of variants of the same compression method can be good direction. An example is trying to synchronize unstructured and structured pruning. Such methods can be possible with Reinforcement Learning (RL) based approach pursue as the algorithms get more faster and advanced. There is also a lack of exploration combining other compression methods. Additionally, a comprehensive survey of this specific area can also be helpful for researchers.

### 5.3 Model Compression Beyond Size Reduction

Despite efforts to train smaller networks from scratch [19], most Model Compression methods are applied after a big model is trained. Thus, the compression ought to have an effect on other aspects of the model. For example, as pointed out in [13], there are still remaining works that can bridge the concept of a models size (compression) and its Explainability. Some applications of Explainability are mentioned in the pruning section of this writing. But, there are only few works,save for [11], that even acknowledge the existence of a bridge between them. Thus, a formalized future work in this direction can be fruitful.

Basically, one can raise many components of the network and ask how compression impacts it. Presented in section four are only a limited number of them

because there is still much work to be done in the area. Again, a comprehensive survey on the effect of Model Compression beyond size reduction can be extremely helpful to build Model Compression without ramifications and beyond.

### 5.4 Miscellaneous

Some model compression methods are developed and are experimented for a specific types of architectures. These makes them limited in applications. In general Reinforcement Learning (RL) based approaches will be suitable in the future to adapt compression techniques to the given model.

There is little innovation in methods other than pruning, quantization and knowledge distillation. Design choices can make a considerable change [66], [80], [34] [35]. Since the aim problem of model compression is simple and intuitive, any model size reduction method can be called a model compression technique as long as it performs well.

The idea of ensembles have been in use and in classical Machine Learning (ML) world. It is about using multiple machine learning models for a single task simultaneously. It increases accuracy by taking advantage of the performance of individual models. The concept have been far from being even mentioned in the context of neural networks as the complexity is overwhelming even to thing about. But with parallel advances in model compression and hardware, it seems it will be within the reach of industry leaders or whoever has the access and motive to try. Since that has most likely never been explored, it is worth the shot. Hardware issues, thought they are not discussed in this survey, are highly related with compression and optimization.

## 6 Conclusion

This paper presents a brief review of the state-of-the-art techniques, methods, challenges, and opportunities pertaining to machine learning model compression. The comprehensive reviewed methods output methods will serve as a stepping stone for future research. It started by defining compression and its challenges. Then thirty years of work on pruning, a number of works on knowledge distillation and quantization are synthesized to yield this work. It included findings of each method under their section. It contained a dedicated section intended to magnify the relationship between model compression and other architectural endeavours. The survey finalized by a discussion section that summarized the patterns in the literature. Actionable future works, that have never been mentioned in literature, have been put forward to assist interested researchers.

## References

1. Aghli, N., Ribeiro, E.: Combining weight pruning and knowledge distillation for cnn compression. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 3185–3192 (2021)

2. Ashok, A., Rhinehart, N., Beainy, F.N., Kitani, K.M.: N2n learning: Network to network compression via policy gradient reinforcement learning. ArXiv **abs/1709.06030** (2018)
3. Bang, D., Lee, J., Shim, H.: Distilling from professors: Enhancing the knowledge distillation of teachers. Information Sciences **576**, 743–755 (2021). https://doi.org/https://doi.org/10.1016/j.ins.2021.08.020, https://www.sciencedirect.com/science/article/pii/S0020025521008203
4. Banner, R., Nahshan, Y., Hoffer, E., Soudry, D.: Aciq: Analytical clipping for integer quantization of neural networks. ArXiv **abs/1810.05723** (2018)
5. Bengio, Y., et al.: Learning deep architectures for ai. Foundations and trends® in Machine Learning **2**(1), 1–127 (2009)
6. Bianco, S., Cadene, R., Celona, L., Napoletano, P.: Benchmark analysis of representative deep neural network architectures. IEEE Access **6**, 64270–64277 (2018). https://doi.org/10.1109/ACCESS.2018.2877890
7. Blalock, D.W., Ortiz, J.J.G., Frankle, J., Guttag, J.V.: What is the state of neural network pruning? ArXiv **abs/2003.03033** (2020)
8. Boo, Y., Shin, S., Choi, J., Sung, W.: Stochastic precision ensemble: self-knowledge distillation for quantized deep neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 6794–6802 (2021)
9. Bucila, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: KDD '06 (2006)
10. Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Zeroq: A novel zero shot quantization framework. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 13166–13175 (2020)
11. Calvi, G.G., Moniri, A., Mahfouz, M., Zhao, Q., Mandic, D.P.: Compression and interpretability of deep neural networks via tucker tensor layer: From first principles to tensor valued back-propagation. arXiv: Learning (2019)
12. Chen, L., Chen, Y., Xi, J., Le, X.: Knowledge from the original network: restore a better pruned network with knowledge distillation. Complex & Intelligent Systems (2021)
13. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. ArXiv **abs/1710.09282** (2017)
14. Courbariaux, M., Bengio, Y.: Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. ArXiv **abs/1602.02830** (2016)
15. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: NIPS (2015)
16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. ArXiv **abs/1810.04805** (2019)
17. Ding, X., Wang, Y., Xu, Z., Wang, Z.J., Welch, W.J.: Distilling and transferring knowledge via cgan-generated samples for image classification and regression. Expert Systems with Applications **213**, 119060 (2023). https://doi.org/https://doi.org/10.1016/j.eswa.2022.119060, https://www.sciencedirect.com/science/article/pii/S0957417422020784
18. Dong, X., Yang, Y.: Network pruning via transformable architecture search. In: NeurIPS (2019)
19. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv: Learning (2019)
20. Frosst, N., Hinton, G.E.: Distilling a neural network into a soft decision tree. ArXiv **abs/1711.09784** (2017)
21. Furlanello, T., Lipton, Z.C., Tschannen, M., Itti, L., Anandkumar, A.: Born again neural networks. In: ICML (2018)

22. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. ArXiv **abs/2103.13630** (2022)
23. Gong, Y., Liu, L., Yang, M., Bourdev, L.D.: Compressing deep convolutional networks using vector quantization. ArXiv **abs/1412.6115** (2014)
24. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. ArXiv **abs/2006.05525** (2021)
25. Gupta, M., Aravindan, S., Kalisz, A., Chandrasekhar, V.R., Jie, L.: Learning to prune deep neural networks via reinforcement learning. ArXiv **abs/2007.04756** (2020)
26. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. arXiv: Computer Vision and Pattern Recognition (2016)
27. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015)
28. Hassibi, B., Stork, D., Wolff, G.: Optimal brain surgeon and general network pruning. In: IEEE International Conference on Neural Networks. pp. 293–299 vol.1 (1993). https://doi.org/10.1109/ICNN.1993.298572
29. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. ArXiv **abs/1808.06866** (2018)
30. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S.: Amc: Automl for model compression and acceleration on mobile devices. In: ECCV (2018)
31. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 1398–1406 (2017)
32. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. ArXiv **abs/1503.02531** (2015)
33. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
34. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv **abs/1704.04861** (2017)
35. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡1mb model size. ArXiv **abs/1602.07360** (2016)
36. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A.G., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 2704–2713 (2018)
37. Joseph, V., Siddiqui, S.A., Bhaskara, A., Gopalakrishnan, G., Muralidharan, S., Garland, M., Ahmed, S., Dengel, A.R.: Going beyond classification accuracy metrics in model compression (2020)
38. Kim, J., Bhalgat, Y., Lee, J., Patel, C., Kwak, N.: Qkd: Quantization-aware knowledge distillation. arXiv preprint arXiv:1911.12491 (2019)
39. Kim, Y., Rush, A.M.: Sequence-level knowledge distillation. In: Conference on Empirical Methods in Natural Language Processing (2016)
40. Kossaifi, J., Lipton, Z.C., Khanna, A., Furlanello, T., Anandkumar, A.: Tensor regression networks. ArXiv **abs/1707.08308** (2020)
41. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**, 84 – 90 (2012)

42. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: NIPS (1989)
43. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. ArXiv **abs/1608.08710** (2017)
44. Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., Gonzalez, J.: Train large, then compress: Rethinking model size for efficient training and inference of transformers. ArXiv **abs/2002.11794** (2020)
45. Liu, X., Wang, X., Matwin, S.: Improving the interpretability of deep neural networks with knowledge distillation. 2018 IEEE International Conference on Data Mining Workshops (ICDMW) pp. 905–912 (2018)
46. Malach, E., Yehudai, G., Shalev-Shwartz, S., Shamir, O.: Proving the lottery ticket hypothesis: Pruning is all you need. In: ICML (2020)
47. Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant. In: AAAI (2020)
48. Mozer, M.C., Smolensky, P.: Skeletonization: A technique for trimming the fat from a network via relevance assessment. In: NIPS (1988)
49. Müller, R., Kornblith, S., Hinton, G.E.: Subclass distillation. ArXiv **abs/2002.03936** (2020)
50. Nayak, G.K., Mopuri, K.R., Shaj, V., Babu, R.V., Chakraborty, A.: Zero-shot knowledge distillation in deep networks. ArXiv **abs/1905.08114** (2019)
51. Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D.P.: Tensorizing neural networks. In: NIPS (2015)
52. Olah, C.: Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases. https://transformer-circuits.pub/2022/mech-interp-essay/index.html (jun 22 2022), [Online; accessed 2022-12-14]
53. Park, G., Yang, J.Y., Hwang, S.J., Yang, E.: Attribution preservation in network compression for reliable network interpretation. Advances in Neural Information Processing Systems **33**, 5093–5104 (2020)
54. Park, J., No, A.: Prune your model before distill it. In: European Conference on Computer Vision (2021)
55. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 3962–3971 (2019)
56. Patterson, D., Gonzalez, J., Le, Q.V., Liang, C., Munguía, L.M., Rothchild, D., So, D.R., Texier, M., Dean, J.: Carbon emissions and large neural network training. ArXiv **abs/2104.10350** (2021)
57. Persand, K., Anderson, A., Gregg, D.: Taxonomy of saliency metrics for channel pruning. IEEE Access **9**, 120110–120126 (2021). https://doi.org/10.1109/ACCESS.2021.3108545
58. Polino, A., Pascanu, R., Alistarh, D.: Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668 (2018)
59. Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., Rastegari, M.: What's hidden in a randomly weighted neural network? 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 11890–11899 (2020)
60. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: ECCV (2016)
61. Reed, R.: Pruning algorithms-a survey. IEEE Transactions on Neural Networks **4**(5), 740–747 (1993). https://doi.org/10.1109/72.248452
62. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. CoRR **abs/1412.6550** (2015)
63. Sau, B.B., Balasubramanian, V.N.: Deep model compression: Distilling knowledge from noisy teachers. arXiv preprint arXiv:1610.09650 (2016)

64. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2015)
65. Stoychev, S., Gunes, H.: The effect of model compression on fairness in facial expression recognition. ArXiv **abs/2201.01709** (2022)
66. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1–9 (2015)
67. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. ArXiv **abs/1910.10699** (2020)
68. Van Baalen, M., Louizos, C., Nagel, M., Amjad, R.A., Wang, Y., Blankevoort, T., Welling, M.: Bayesian bits: Unifying quantization and pruning. Advances in neural information processing systems **33**, 5741–5752 (2020)
69. Wang, Y., Lu, Y., Blankevoort, T.: Differentiable joint pruning and quantization for hardware efficiency. ArXiv **abs/2007.10463** (2020)
70. Wang, Y., Zhang, X., Hu, X., Zhang, B., Su, H.: Dynamic network pruning with interpretable layerwise channel selection. In: AAAI (2020)
71. Wang, Y., Zhang, X., Xie, L., Zhou, J., Su, H., Zhang, B., Hu, X.: Pruning from scratch. In: AAAI (2020)
72. Wang, Z.: Data-free knowledge distillation with soft targeted transfer set synthesis. In: AAAI (2021)
73. Wiedemann, S., Kirchhoffer, H., Matlage, S., Haase, P., Marbán, A., Marinc, T., Neumann, D., Nguyen, T., Schwarz, H., Wiegand, T., Marpe, D., Samek, W.: Deepcabac: A universal compression algorithm for deep neural networks. IEEE Journal of Selected Topics in Signal Processing **14**, 700–714 (2020)
74. Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., Xu, C.: Cars: Continuous evolution for efficient neural architecture search. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1826–1835 (2020)
75. Yao, K., Cao, F., Leung, Y.W., Liang, J.: Deep neural network compression through interpretability-based filter pruning. Pattern Recognit. **119**, 108056 (2021)
76. Yeom, S.K., Seegerer, P., Lapuschkin, S., Wiedemann, S., Müller, K.R., Samek, W.: Pruning by explaining: A novel criterion for deep neural network pruning. ArXiv **abs/1912.08881** (2021)
77. Yoo, J., Cho, M., Kim, T., Kang, U.: Knowledge extraction with no observable data. In: NeurIPS (2019)
78. Yu, R., Li, A., Chen, C.F., Lai, J.H., Morariu, V.I., Han, X., Gao, M., Lin, C.Y., Davis, L.S.: Nisp: Pruning networks using neuron importance score propagation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 9194–9203 (2018)
79. Yuan, L., Tay, F.E., Li, G., Wang, T., Feng, J.: Revisiting knowledge distillation via label smoothing regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3903–3911 (2020)
80. Zhai, S., Cheng, Y., Zhang, Z., Lu, W.: Doubly convolutional neural networks. In: NIPS (2016)
81. Zhang, Z., Shao, W., Gu, J., Wang, X., Ping, L.: Differentiable dynamic quantization with mixed precision and adaptive resolution. ArXiv **abs/2106.02295** (2021)