Distilled Gradual Pruning With Pruned Fine-Tuning

Federico Fontana, *Student Member, IEEE*, Romeo Lanzino, *Student Member, IEEE*, Marco Raoul Marini, *Member, IEEE*, Danilo Avola, *Member, IEEE*, Luigi Cinque, *Senior Member, IEEE*, Francesco Scarcello, and Gian Luca Foresti, *Senior Member, IEEE*

***Abstract*—Neural networks (NNs) have been driving machine learning progress in recent years, but their larger models present challenges in resource-limited environments. Weight pruning reduces the computational demand, often with performance degradation and long training procedures. This work introduces distilled gradual pruning with pruned fine-tuning (DG2PF), a comprehensive algorithm that iteratively prunes pretrained NNs using knowledge distillation. We employ a magnitude-based unstructured pruning function that selectively removes a specified proportion of unimportant weights from the network. This function also leads to an efficient compression of the model size while minimizing classification accuracy loss. Additionally, we introduce a simulated pruning strategy with the same effects of weight recovery but while maintaining stable convergence. Furthermore, we propose a multistep self-knowledge distillation strategy to effectively transfer the knowledge of the full, unpruned network to the pruned counterpart. We validate the performance of our algorithm through extensive experimentation on diverse benchmark datasets, including CIFAR-10 and ImageNet, as well as a set of model architectures. The results highlight how our algorithm prunes and optimizes pretrained NNs without substantially degrading their classification accuracy while delivering significantly faster and more compact models.**

***Impact Statement*—In recent times, NNs have demonstrated remarkable outcomes in various tasks. Some of the most advanced possess billions of trainable parameters, making their training and inference both energy intensive and costly. As a result, the focus on pruning is growing in response to the escalating demand for NNs. However, most current pruning techniques involve training a model from scratch or with a lengthy training process leading to a significant increase in carbon footprint, and some experience a notable drop in performance. In this article, we introduce DG2PF. This unstructured pruning algorithm operates on pretrained NNs, allows the user to choose the proportion of parameters to prune, and halts automatically when the pruned network has achieved optimal performance, thereby preventing excessive training time. We envision that with DG2PF even the most sophisticated new NNs could become accessible to the average user.**

***Index Terms*—Artificial intelligence in computational sustainability, deep learning, neural networks (NNs), supervised learning.**

Manuscript received 27 June 2023; revised 25 September 2023 and 23 November 2023; accepted 10 February 2024. Date of publication 15 February 2024; date of current version 13 August 2024. This work was supported in part by the "Smart unmannEd AeRial vehiCles for Human likE monitoRing (SEARCHER)" project of the Italian Ministry of Defence under Grant CIG: Z84333EA0D and in part by the PNRR project FAIR—Future AI Research through the NRRP MUR program funded by NextGenerationEU under Grant PE00000013. This article was recommended for publication by Associate Editor Jingfeng Zhang upon evaluation of the reviewers' comments. *(Corresponding author: Romeo Lanzino.)*

Federico Fontana, Romeo Lanzino, Marco Raoul Marini, Danilo Avola, and Luigi Cinque are with the Department of Computer Science, Sapienza University of Rome, 00198 Rome, Italy (e-mail: fontana.f@di.uniroma1.it; lanzino@di.uniroma1.it; marini@di.uniroma1.it; avola@di.uniroma1.it; cinque@di.uniroma1.it).

Francesco Scarcello is with the Department of Computer Engineering, Modeling, Electronics and Systems, University of Calabria, 87030 Rende, Italy (e-mail: scarcello@dimes.unical.it).

Gian Luca Foresti is with the Department of Mathematics, Computer Science and Physics, University of Udine, 33100 Udine, Italy (e-mail: gianluca.foresti@uniud.it).

Digital Object Identifier 10.1109/TAI.2024.3366497

I. Introduction

Deep neural networks (NNs) have shown state-of-the-art performance on various visual tasks, such as image classification [1], [2], [3], [4], object detection [5], [6], and semantic segmentation [7], [8]. Despite their success, the substantial size and computational demands of these models present a major challenge for their implementation on resource-limited devices. Several compression techniques have been developed to reduce the size and computational demands of deep NNs while retaining their performance and to overcome the previously mentioned challenges. Neural architecture search (NAS) has been explored as a method to design efficient architectures; for instance, in [9] an optimization for specific hardware platforms is proposed, and in [10] the curriculum search strategy is explored. They support the expansion of the search space progressively. Techniques such as the contrastive learning framework [11], the "once-for-all" approach [12], and the neural architecture Transformer [13] have further advanced the field. Last, the disturbance-immune update strategy [14] addresses the performance disturbance issue in weight-sharing NAS methods. However, while NAS offers automated design, the need for more direct compression techniques remains paramount. This is where pruning comes into play. This work delves deeper into the intricacies and advancements in pruning techniques.

The primary goal of weight pruning is to remove nonrelevant weights from a NN. This process aims to reduce the network's size and computational requirements while minimizing the loss of its performance. There are two types of pruning methods, structured and unstructured. Structured pruning involves modifying or removing layers or parts of the network. This method may lead to changes in the input and output dimensions of the layers, which can cause issues in networks with long-range dependencies among layers [15]. The solution to this problem is often circumvented by constraining pruning into targeting only layers that do not induce issues like filters [16] and channels pruning [17], [18], or a mixed approach [19]. Whatever the pruning method be, it usually involves careful fine-tuning [20] to maximize its performances. However, such constraints are expected to decrease the efficiency of pruning. Unstructured pruning, on the other hand, produces sparse matrices that are difficult to accelerate [21], even if some recent works withdraw this statement [22], [23]. In this context, different strategies have been proposed throughout the years for unstructured pruning in several application areas. The optimal brain damage algorithm [24] and magnitude-based pruning algorithm [25] are two popular unstructured pruning techniques. Other popular methods include Taylor expansion pruning [26], which prunes based on the loss function's second-order Taylor approximation, and random pruning [25], which prunes randomly to improve computation times. However, a simple pruning of the weights may lead to a drop in performance. To this extent, weight recovery between training cycles [27], [28] and fine-tuning the pruned model through additional training has shown to be an effective approach to mitigate this issue [29].

As a popular approach for model compression, knowledge distillation has received significant attention in recent years [30], [31]. The basic idea behind this technique is to train a smaller model, referred to as the student model, and to mimic the behavior of a larger model, referred to as the teacher model. The student model is trained by minimizing the difference between its predictions and the predictions of the teacher model, which is often a pretrained NN. Self-distillation refers to a knowledge distillation approach where a NN is distilled into a smaller, more compact version of itself [32].

The research direction goes toward more complex pruning and distillation strategies, but often with a large computational cost; Srinivas et al. [28] tried to introduce a cyclical pruning and weight recovery schedule, but significantly increasing the complexity of the algorithm at a price of a slight classification improvement.

We present a novel unstructured pruning algorithm that seamlessly integrates knowledge distillation techniques to achieve significant model compression without compromising its accuracy. Our proposed method commences with a gradual weight pruning phase that employs knowledge distillation to remove unimportant weights and reduce the model size. Once the desired sparsity level is achieved, the model undergoes a distilled fine-tuning process until convergence. This is then followed by a final fine-tuning process without the teacher. We demonstrate that our approach outperforms the existing methods in terms of compression-accuracy tradeoffs through extensive experimental evaluations conducted on publicly available benchmark datasets. These results show that the algorithm has a potential impact in the field of deep learning by enabling the deployment of large, accurate models on a wide range of devices with limited computational resources and to average users.

To summarize, the contributions of this work are as follows.

1) We build upon a well-known baseline function exploiting magnitude-based unstructured pruning to minimize memory and storage requirements by selectively removing a specified proportion of weights from a pretrained NN.
2) We propose a unique simulated pruning technique. This method stands out as it replicates the benefits of weight recovery while consistently maintaining stable convergence. Notably, this is achieved at each training iteration, setting it apart from conventional practices in weight recovery literature.
3) We introduce distilled gradual pruning with pruned fine-tuning (DG2PF), a comprehensive algorithm that integrates unstructured weight pruning and knowledge distillation to prune pretrained NNs without incurring a substantial reduction in performance.
4) We have conducted experiments on publicly available benchmark datasets and models to validate the performance of our method. The results of this evaluation provide quantifiable evidence of the effectiveness of the proposed algorithm.

The rest of this article is organized as follows: Section II presents the related work, where we review and discuss the existing literature and research relevant to our study; Section III contains details about the proposed algorithm, comprehensive of pseudocode; Section IV details the evaluation of DG2PF and the comparative studies with the state-of-the-art pruning techniques on two representative datasets; and Section V discusses and presents the conclusion and future work.

II. Related Work

Pruning in NNs can involve either structured pruning that removes model structures or unstructured pruning that removes individual parameters. In general, structured pruning methods [16], [33] do not depend on specialized hardware. In contrast, unstructured pruning approaches [34], [35] explicitly require support for sparse computations. Recent advancements in structured pruning include [17], which aims to enhance network performance through channel pruning by eliminating redundant components. The work in [18] offers a distinctive method for lossless channel pruning, drawing inspiration from neurobiology, and ensures structured sparsity without sacrificing accuracy. Meanwhile, Liu et al. [36] introduce a combined approach of discrimination-aware channel and kernel pruning. In the context of unstructured pruning, there are three distinct pruning schedules: one-shot, gradual, and cyclical pruning. One-shot pruning [37] involves the simultaneous removal of unimportant weights in a single step, followed by a final fine-tuning stage. Gradual pruning [27] gradually prunes the network weights over multiple iterations. This approach is interleaved with training steps and culminates in a final fine-tuning stage. Cyclical pruning [28] involves multiple gradual pruning schedules, with weight recovery at the beginning of each cycle. Parameter-efficient masking networks [38] lead to a new paradigm for model compression utilizing one random initialized layer, accompanied by different masks, so the model can be expressed as one-layer with a bunch of masks. The work in [39] smoothly induces sparsity while learning pruning thresholds, providing a nonuniform sparsity budget.

This article, inspired by [27], proposes an algorithm that fuses pruning and knowledge distillation techniques, introducing a novel approach called simulated pruning. The simulated pruning introduces weight recovery without the need for cyclical schedules. In [40] and [41], the authors suggest automatically tuning thresholds for magnitude pruning to improve global sparsity by removing unimportant weights based on their absolute value. Alternative approaches to magnitude pruning, such as second-order [24], [42] and Fisher-based [43], [44] of the loss function, have been proposed. However, recent work [45] suggests they may not be more effective, especially when combined with fine-tuning. Probabilistic pruning approaches, such as those described in [46] and [47], involve stochastic relaxations, but research [48] shows they often perform similarly to simple magnitude pruning-based methods. The works described in [49] and [50] use gradient updates computed on a sparse proxy model by exploiting the straight-through estimator (STE), similar to [51] and [52], and claim that this method can lead to weight recovery. These approaches make use of one-shot pruning. However, Srinivas et al. [28] show weight recovery is complicated to achieve in practice in this setting.

Knowledge distillation is a form of compression strategy that transfers relevant feature representation from a larger teacher network to a smaller student network, followed by fine-tuning. This method was proposed by [53] for networks that tackle the classification task. The approach introduces a distillation loss that utilizes the softened output of the teacher network's last layer. In [30], the authors improved the performance of this approach by using an intermediate representation of the teacher model as a hint in addition to the output layer. In [54], knowledge distillation is applied to the ResNet architecture by minimizing the L_2 loss of the Gramian feature matrix in the ResNet modules between teacher and student. Like for our article, recent works [55], [56] try to mix pruning and distillation for optimal performance.

III. Proposed Method: DG2PF

In this section, we will describe our proposed method, called DG2PF. The algorithm is composed of two phases. The first phase, called distilled gradual pruning (DGP) (Algorithm 1), incorporates two distinct types of pruning mechanisms. The first type of pruning is carried out according to the procedure outlined in Section III.A. This pruning approach is gradually applied, once per epoch, during the first phase, until the desired sparsity level is attained. We called the other kind of pruning "simulated," as described in Section III.B. This type of pruning is performed during each iteration of every epoch of the DGP phase. It selectively removes and recovers a portion of the weights that have not yet been pruned in the network. The second phase is called pruned fine-tuning (PF) (Algorithm 2) and starts upon completion of the previous one. Here, the network has already been pruned to its intended sparsity level and the simulated pruning strategy is terminated. This phase aims at recovering most of the performance lost during DGP. In Section III.C, a knowledge distillation strategy is presented. It merges two knowledge distillation losses, named Kullback–Leibler (KL) divergence and performance-weighted loss. In Section III.D, we present DG2PF, our novel two-phase algorithm that merges the techniques mentioned above.

Algorithm 1 Distilled Gradual Pruning
```
$i \leftarrow 1$
$\boldsymbol{\delta} \leftarrow$ linearly sample $s_e$ numbers in $[0, s]$
while $i \leq s_e$ or the score keeps improving do
    if $i \leq s_e$ then
        prune $\boldsymbol{\delta}_i$ percent of the model
    end if
    for $b \in \mathcal{D}_t^{(b)}$ do
        if $i \leq s_e$ then
            apply simulated pruning to the weights (2)
        end if
        $\boldsymbol{y} \leftarrow$ ground truth labels for the $b$-th batch
        $\hat{\boldsymbol{y}} \leftarrow$ model's predictions for the $b$-th batch
        $\hat{\boldsymbol{y}}^{(t)} \leftarrow$ teacher's predictions for the $b$-th batch
        $\mathcal{L} \leftarrow$ KD loss (9)
        $\Delta\theta \leftarrow$ gradients from $\mathcal{L}$
        if $i \leq s_e$ then
            recover the weights of the simulated pruning
        end if
        update unpruned weights with $\Delta\theta$ using AdamW
    end for
    score $\leftarrow$ top-1 validation accuracy (10) on $\mathcal{D}_v^{(b)}$
    $i \leftarrow i + 1$
end while
```

Algorithm 2 Pruned Fine-tuning
```
while the score keeps improving do
    for $b \in \mathcal{D}_t^{(b)}$ do
        $\boldsymbol{y} \leftarrow$ ground truth labels for the $b$-th batch
        $\hat{\boldsymbol{y}} \leftarrow$ model's predictions for the $b$-th batch
        $\mathcal{L} \leftarrow$ CE loss (4)
        $\Delta\theta \leftarrow$ gradients from $\mathcal{L}$
        update unpruned weights with $\Delta\theta$ using SGD
    end for
    score $\leftarrow$ top-1 validation accuracy (10) on $\mathcal{D}_v^{(b)}$
end while
```

A. Pruning Function

In line with the previous research [40], [41], we operate with the assumption that weights with magnitudes closer to zero have less impact on the final output of a NN. Therefore, we propose to prune these weights by collapsing them to zero and flagging them as pruned [27], [28], [37]. The rationale behind this assumption is that the weights with smaller magnitudes have minor effects on the output of the NN. It can be deducted by considering the activation functions commonly used in NNs. In these activation functions, the signal is passed through a hard or soft threshold, which means that small changes in the input signal do not or marginally affect the output unless they cross this threshold. Thus, weights with smaller magnitudes have a lower probability of crossing the threshold and therefore are less influential in determining the final output. Based on these assumptions, we can remove the

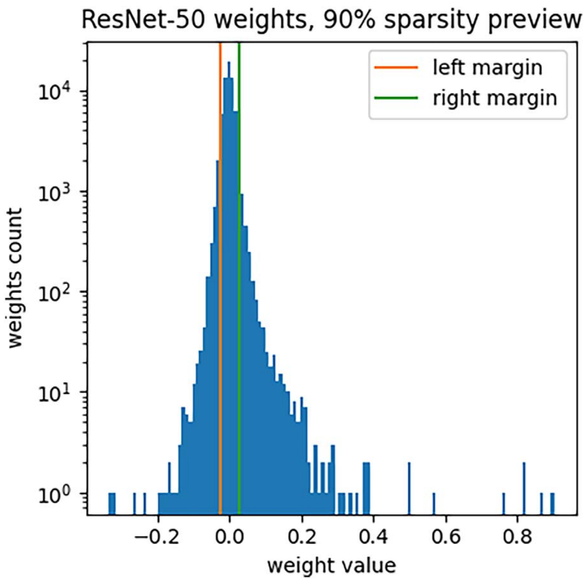

Fig. 1. Histogram representation of the 90% of weights that would be pruned on an unpruned ResNet-50 model [3]. The abscissa depicts the values of the weights, while the ordinate depicts the frequency count of weights with the corresponding value. The vertical bars represent the left and right margins, respectively. The amount of the margin delimits the weights to the p-percentile of the total weights, where p is the arbitrary percentage of pruning set to 0.9 in the plot.

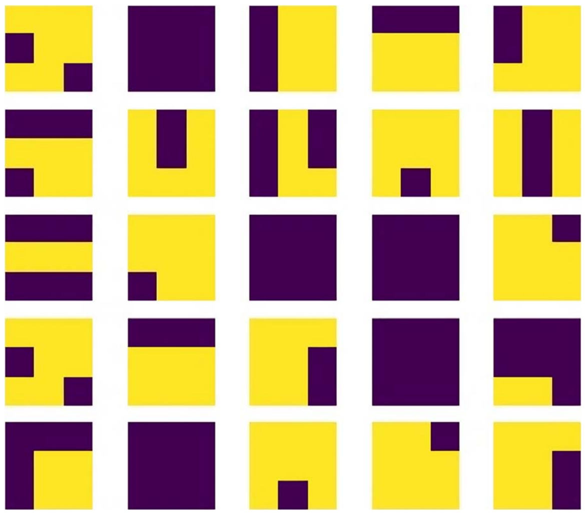

Fig. 2. Illustration of pruned and unpruned parameters within a layer of a 70% sparse MobileNet V2 network. Each 3×3 matrix depicts a channel of the layer's weights. Within each filter, the pruned parameters are shaded in a darker tone, whereas the unpruned parameters are highlighted in yellow.

weights with smaller magnitudes without a significant loss of accuracy. Consequently, the number of parameters in the network is reduced, improving its efficiency without significant performance degradation.

Let $s \in \mathbb{R}$ be the chosen sparsity of the network, with $0 < s < 1$. Each weight θ_i of a NN parameterized by θ is pruned as follows:

$$\theta_i = \begin{cases} \theta_i, & \text{if } \theta_i < m_l \text{ and } \theta_i > m_r \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where m_l and $m_r \in \mathbb{R}$ are the margins computed as $(1 - s/2)$th and $(s + (1 - s/2))$th percentiles of the weights θ, respectively. The weights falling inside these margins are set to zero and thus pruned. Fig. 1 shows an example of margins and weights to prune on a pretrained network. Moreover, Fig. 2 depicts an example of pruned and unpruned parameters within a layer of a sparse network.

B. Simulated Pruning Function

We assume that the reduction of the importance of weights likely to become zero during the upcoming pruning stage has a comparatively minor impact on the network's overall performance. When we employ this technique, we essentially carry out a cyclical pruning step in a single training iteration on a single batch of data. It means that in each iteration we start with the (simulated) pruning stage and we recover the pruned weights by the end of the iteration. This methodology stands in contrast to the approach presented in [28], where the pruning process is initiated only after completing a predetermined number of training epochs. In particular, in [28] each cycle spans several training epochs and ensures that weights undergo a gradual pruning, in order to only have a fraction restored at the end of the cycle. A notable limitation emerges when these weights, especially in the earlier stages of the cycle, are pruned based on a constrained pool of information. It predominantly happens when specific policies, such as magnitude-based pruning, are adopted. Despite the evident efficacy of the cyclical pruning mechanism, our methodology compares and rectifies its core shortcomings. We guarantee that the heuristic responsible for the pruning decision is perpetually equipped with a uniform dataset for each weight, facilitating both the pruning and recovery within each iteration, ensuring an informed decision-making process, and enabling more stable convergence. We use straight through estimation (STE), thus allowing the gradient to pass through the weights pruned in this phase. As theoretically proved by [57], this technique speeds up the learning process and helps ensure stability.

Let $s_{\text{sim}} \in \mathbb{R}$ be the chosen simulated sparsity of the network, with $0 < s < 1$. At the start of each training step of the first phase of the algorithm, a fraction s_{sim} of unpruned weights are pruned and then recovered after the backpropagation of the loss. Each weight θ_i of a NN parameterized by θ is pruned according to the probability

$$\theta_i = \begin{cases} \theta_i, & \text{if } \boldsymbol{p}_i < m_s \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where m_s corresponds to the $(1 - s_{\text{sim}})$th percentile of a vector $\boldsymbol{p} \in [0, 1]^{|\theta|}$ obtained as follows:

$$\boldsymbol{p} = \mathbb{1} - \frac{\text{abs}(\theta)}{\max(\text{abs}(\theta))} \quad (3)$$

where $\mathbb{1} = [1]^{|\theta|}$ is a vector of the same length of θ where each position is filled with 1.

C. Knowledge Distillation Procedure

As will be described in Section III.D, the proposed method follows two phases. The first one entails knowledge distillation from the original, unpruned model. The loss used to train the student model during these steps combines a variation of performance-weighted loss [58] and pointwise KL divergence loss [59].

The rationale under their combination is to use performance-weighted loss to get resilience against outliers and challenging instances, and pointwise KL divergence to align the student model's distribution with the teacher model's distribution.

The performance-weighted loss is a modification of the well-known cross-entropy loss. The cross-entropy loss is commonly used for training classification networks and is expressed mathematically as

$$\mathcal{L}_{\text{CE}} = -\frac{1}{B}\sum_{i=1}^{B} \boldsymbol{y}_i \log(\hat{\boldsymbol{y}}_i) \tag{4}$$

where B is the batch size, $\boldsymbol{y}_i$ is the ground truth label vector for the ith sample, and $\hat{\boldsymbol{y}}_i$ contains the predicted probabilities for sample i. The logarithm in the formula is used to amplify the loss when the model is highly confident but incorrect. In fact, the logarithm grows as the predicted probability approaches 0, penalizing the model for being overly confident in incorrect predictions. As a more robust alternative to the cross-entropy loss, during the distillation phase of the algorithm, an alternative version of the performance-weighted loss [58] is employed. In this procedure, each sample is given a proportional weight to the teacher network's confidence when classifying. Thus, the weight w_i of a sample of index i in a batch is defined starting from the score of the teacher network for the correct class c_i, $\hat{\boldsymbol{y}}^{(t)}_{i,c_i} \in \mathbb{R}$, as follows:

$$w_i = (1 - \hat{\boldsymbol{y}}^{(t)}_{i,c_i})^{\gamma} + \beta \tag{5}$$

with $\gamma > 0$ set to 1 and $\beta \in [0, 1]$ set to 0.1. Since (5) puts more emphasis on incorrect labels, the original authors propose to compare student network's predictions to corrected soft-labels $\hat{\boldsymbol{y}}^{*}_i$ instead of always the ground truth labels $\boldsymbol{y}_i$

$$\hat{\boldsymbol{y}}^{*}_i = \begin{cases} \hat{\boldsymbol{y}}_i, & \text{if the sample is correctly classified} \\ \boldsymbol{y}_i, & \text{otherwise} \end{cases} \tag{6}$$

where student network's predictions $\hat{\boldsymbol{y}}_i$ are used instead of the one-hot encoded ground truth vector $\boldsymbol{y}_i$ where the model has made a correct classification. Given that the modified performance-weighted loss is defined as

$$\mathcal{L}_{\text{PW}} = \frac{1}{B}\sum_{i=1}^{B} w_i \cdot \mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}^{*}_i, \hat{\boldsymbol{y}}_i) \tag{7}$$

where B is the batch size, $\mathcal{L}_{\text{CE}}$ is the cross-entropy function (4), w_i is the weight of the ith sample in the batch (5), and $\hat{\boldsymbol{y}}^{*}_i$ is the corrected soft-labels vector (6).

The pointwise KL divergence loss measures the dissimilarity between two probability distributions. It is commonly used in knowledge distillation to match the soft predictions of a more extensive, pretrained teacher network to those of a smaller student network [53], [59]. The formula for the pointwise KL loss is defined as follows:

$$\mathcal{L}_{\text{KL}} = \frac{1}{B}\sum_{i=1}^{B} \boldsymbol{y}^{(t)}_i \cdot (\log(\boldsymbol{y}^{(t)}_i) - \boldsymbol{y}_i) \tag{8}$$

where B is the batch size, while $\boldsymbol{y}^{(t)}_i$ and $\boldsymbol{y}_i$, respectively, contain the predictions of the teacher and the student networks on the ith sample.

The final loss function utilized in the first two stages of the procedure is a modified version of the one proposed in a previous study [53], which is calculated as follows:

$$\mathcal{L}_{\text{KD}} = (\alpha \cdot \mathcal{L}_{\text{KL}} + (1-\alpha) \cdot \mathcal{L}_{\text{PW}}) \cdot \tau^2. \tag{9}$$

In this equation, $\mathcal{L}_{\text{KD}}$ emerges as a linear combination of the two sublosses $\mathcal{L}_{\text{KL}}$ (8) and $\mathcal{L}_{\text{PW}}$ (7), modulated by parameters $\alpha \in [0, 1]$ and $\tau \in \mathbb{R}$. The coefficient α acts as a balancing factor, determining the proportional influence of $\mathcal{L}_{\text{KL}}$ on the overall loss. Meanwhile, τ functions as a temperature parameter. Notably, the combination is weighted by τ^2, thereby adjusting the scale and sensitivity of the combined loss. In a broader sense, α and τ adjust the balance and sensitivity of the loss function, determining the importance of replicating the teacher network's behavior via $\mathcal{L}_{\text{KL}}$ and classifying examples through $\mathcal{L}_{\text{PW}}$.

D. Distilled Gradual Pruning With Pruned Fine-Tuning

The proposed algorithm is composed of two phases.

The first phase, called DGP, involves gradually removing parts of the model while minimizing the loss in classification performance. This process is executed by using self-distillation to make sure the pruned model behaves as much like the original model as possible. The algorithm works by gradually pruning the model over a specific number of s_e epochs and then continuing to train until it reaches convergence. The procedure's pseudocode is shown in Algorithm 1. Let $\boldsymbol{\delta} \in [0, s]^{s_e}$ be a vector containing s_e evenly spaced numbers in increasing order. At the beginning of each epoch, $i \leq s_e$ the model is pruned to a sparsity of $\boldsymbol{\delta}_i$ and then trained on the batched training dataset $\mathcal{D}^{(b)}_t$. At the beginning of each training step, the model undergoes an additional simulated pruning process, as explained in Section III.B. This procedure happens only if epoch $i \leq s_e$ and targets the unpruned weights, reducing their sparsity to s_{sim}. Then, the algorithm makes predictions $\hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}^{(t)} \in \mathbb{R}^{b_s \times c}$ using the pruned and teacher models, respectively, where b_s denotes the batch size and c is the number of labels in the datasets. These predictions are compared to the actual labels $\boldsymbol{y} \in \mathbb{R}^{b_s}$, and the knowledge distillation loss $\mathcal{L}$ is calculated using (9). From this loss, we compute the gradients $\Delta\theta$ and eventually restore the weights set to zero during the simulated pruning step. After that, the algorithm updates the unpruned weights and proceeds to the next batch in the epoch. At the end of each training epoch, the model is tested on the batched validation dataset $\mathcal{D}^{(b)}_v$, and its top-1 accuracy score is saved. We use AdamW [60] as the optimizer function to speed up convergence. The DGP process ends when the maximum number of epochs has been reached or if the top-1 accuracy score on the batched validation dataset $\mathcal{D}_v^{(b)}$ does not improve after a fixed number of epochs, triggering an early stop.

The second phase of the algorithm, known as PF, follows the first phase of DGP. In this phase, the model is fine-tuned without a teacher, allowing it to focus on classification scores without being constrained by the teacher's predictions. Additionally, the model is not pruned further as the desired sparsity level was achieved during the previous phase. The pseudocode for PF is provided in Algorithm 2. The algorithm loops through the batches of the training dataset $\mathcal{D}_t^{(b)}$ with the same stopping criteria as the previous phase. During training, the unpruned weights are trained using the cross-entropy loss (4) to enhance classification performance. The unpruned parameters are updated through stochastic gradient descent (SGD) with a low learning rate. We opted for SGD over AdamW since our experiments yielded better generalization performance.

IV. Experimental Results

In this section, we evaluate our proposal on two widely adopted datasets and compare them to several state-of-the-art methods regarding unstructured pruning.

A. Datasets

CIFAR-10 [61] is a small dataset containing 60 000 training images and 10 000 test images, split into ten classes. The images in CIFAR-10 are relatively simple and small, making it a popular dataset for testing algorithms and architectures in their early stages of development.

ImageNet (also known as ImageNet-1K) [62] is a much larger and more complex dataset, containing over 1 million training images and 50 000 validation images, split into 1000 classes. ImageNet offers various classes, from ordinary objects to abstract concepts, e.g., mountains and handwriting. The larger image size of ImageNet provides a more realistic and challenging benchmark for computer vision models.

B. Metrics

The metric we used to quantify the classification performance of a model is the top-k accuracy. When classifying a sample, the model outputs a probability distribution among the possible labels and is trained to give more weight to the more plausible labels. The top-k predictions $\hat{Y}_{i,k}$ for a sample of index i are the labels with the highest scores. This metric measures the proportion of times the model predicts the correct label to be among the top-k predictions

$$\text{accuracy}(k) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1, & \text{if } y_i \in \hat{Y}_{i,k}\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

where N is the number of samples in the dataset, with $1 \leq i \leq N$, y_i is the true label for the ith sample, and $\hat{Y}_{i,k}$ is the set of the top-k predicted labels for the ith sample, with $|\hat{Y}_{i,k}| = k$. According to typical practices in the related literature, we have decided to present the top-1 accuracy results in comparison with the state of the art in Section IV.E.

We assessed the effectiveness of our compression method using the compression rate metric. The compression rate is calculated using the target sparsity, which represents the percentage of weights that are pruned from the original model. The metric is computed as follows:

$$\text{compression rate} = \frac{1}{1-s} \tag{11}$$

where $0 < s < 1$ represents the target sparsity of the network.

C. Implementation Details

The experiments were conducted on a high-performance computer (HPC) equipped with an Nvidia Quadro RTX6000 GPU and 24 GB of VRAM. Minimal data augmentation was applied to ensure a fair comparison with the previous literature [3], [27], [28], [37]. In addition, this procedure also reduces the potential confounding effects that could be introduced by more complex data preprocessing and allows for a more fair and comprehensive evaluation of the impact of the proposed methods. The optimizer used during the self-distillation phase is AdamW [60], with a learning rate of 10^{-5}, β_1 and β_2 equal to 9×10^{-1} and 9.99×10^{-1}, and a weight decay of 10^{-2}. After the teacher is detached from the pruned model, AdamW is replaced with plain SGD with a learning rate of 10^{-4}, a momentum of 9×10^{-1}, and a weight decay of 5×10^{-4}. This optimization swap is motivated by the fact that in our experiments AdamW tended to converge in fewer epochs while SGD has shown better generalization capabilities. We made this change to improve our model's classification performance. During all experiments, the max epochs were set to 100 to be fair in comparison with other works, however thanks to the early stop strategy and AdamW, no experiments reached the max epochs limit.

D. Ablation Study

In this section, we assess the impact of the hyperparameters used in the method's pruning and distillation stages. To conduct the ablation study, we selected the CIFAR-10 dataset [61] and the ResNet-18 model, which are relatively small and enable quicker and more comprehensive evaluation of various combinations of hyperparameters. The model was initially configured with 95% sparsity, 10% simulated sparsity percentage during self-distillation with $\alpha = 0.75$, and ten pruning epochs. We trained and tested the model in this base configuration for each experiment, varying single hyperparameters. Each table row shows the mean and standard deviation of top-1 accuracy obtained from three runs of the same experiment with different seeds. The notation "acc@1" is utilized as an abbreviation for the top-1 accuracy.

1) Effect of s for Sparsity: In this study, we have examined how increasing the target sparsity s of the model affects classification accuracy. The results are presented in Table I. From the results, we can observe that the loss in accuracy is negligible for sparsity values up to 90%, after which the accuracy begins to decline significantly. Specifically, the drop in accuracy from 90% to 95% amounts to 1.65%, which is consistent with the

TABLE I
CLASSIFICATION PERFORMANCES OF RESNET-18 ON CIFAR-10 AT INCREASING TARGET SPARSITY s

s (%)	Params	Flops	acc@1 (%)
20	9.2 M	0.8×	92.80 ± 0.08
40	6.9 M	0.6×	92.79 ± 0.01
60	4.6 M	0.4×	92.78 ± 0.02
80	2.3 M	0.2×	92.78 ± 0.04
90	1.15 M	0.1×	92.91 ± 0.01
95	0.57 M	0.05×	91.26 ± 0.03

TABLE II
CLASSIFICATION PERFORMANCES OF RESNET-18 ON CIFAR-10 AT INCREASING NUMBER OF PRUNING EPOCHS s_e

s_e	acc@1 (%)	s_e (%)	acc@1 (%)
1	90.16 ± 0.45	9	91.14 ± 0.13
3	90.56 ± 0.23	11	91.14 ± 0.14
5	90.43 ± 0.15	13	91.20 ± 0.23
7	91.10 ± 0.04	15	91.49 ± 0.05

TABLE III
CLASSIFICATION PERFORMANCES OF RESNET-18 ON CIFAR-10 AT INCREASING SIMULATED SPARSITY s_{sim}

s_{sim} (%)	acc@1 (%)	s_{sim} (%)	acc@1 (%)
0	90.89 ± 0.56	5	91.17 ± 0.15
1	90.89 ± 0.54	10	91.26 ± 0.03
3	91.22 ± 0.09	20	90.78 ± 0.13

findings of other studies on unstructured pruning [27], [28], [37]. These results demonstrate that while higher sparsity levels can lead to a more compact and efficient model, there is a tradeoff between sparsity and accuracy. Therefore, the target sparsity s should be carefully selected, considering the specific model, dataset, and desired tradeoff between size and accuracy.

2) Number of Pruning Epochs s_e: In this study, we have investigated whether increasing the number of pruning epochs leads to a more accurate model. The results are shown in Table II and demonstrate a clear trend of higher accuracy with increased pruning epochs. The peak gain of 1.33% was observed at $s_e = 15$ compared to the one-shot pruning setting. The gradual and careful selection of the parameters to prune explains this improvement. However, it should be noted that this result may be further improved if the pruning is performed multiple times per epoch, as proved in [28]. However, it is a field of future research and requires further investigation.

3) Simulated Sparsity s_{sim}: The objective of this study was to observe how the network behaves as the percentage of simulated sparsity is increased. The outcomes of the experiments are presented in Table III. The performance of the network without simulated pruning was better than that with 20% simulated sparsity by 0.11% but inferior to that with 10% simulated sparsity by 0.37%. It implies that the simulated sparsity level must be cautiously selected, as a higher level may remove too many parameters, making learning more difficult.

4) Knowledge Distillation α: This study aimed to measure the impact of α in the knowledge distillation loss (9) on the accuracy of the model. The results are in Table IV. The experiments revealed that the best results were achieved

TABLE IV
CLASSIFICATION PERFORMANCES OF RESNET-18 ON CIFAR-10 AT INCREASING DISTILLATION α IN (9)

α (%)	acc@1 (%)	α (%)	acc@1 (%)
0	89.68 ± 0.51	50	91.15 ± 0.19
10	91.10 ± 0.28	75	91.26 ± 0.03
25	91.39 ± 0.07	90	91.42 ± 0.07
		100	91.10 ± 0.21

TABLE V
CLASSIFICATION PERFORMANCES OF RESNET-18 ON CIFAR-10 AT INCREASING TEMPERATURE τ IN (9)

τ	acc@1 (%)	τ (%)	acc@1 (%)
0.1	92.65 ± 0.11	2	92.63 ± 0.07
0.5	92.79 ± 0.08	4	92.70 ± 0.17
1	92.59 ± 0.10	8	92.61 ± 0.10

with α values of 25% and 90%. Specifically, the mean top-1 accuracy was improved by 1.71% and 1.74%, respectively, compared to the undistilled setting. It was observed that generally the experiments with an α greater than 0 showed better mean top-1 accuracy and reduced standard deviation, indicating that the application of knowledge distillation can improve the model's accuracy.

5) Loss Temperature τ: This study aimed to measure the impact of τ in the knowledge distillation loss (9) on the accuracy of the model. The results are in Table V. The experiments revealed that the best result was achieved with a τ value of 0.5, where the mean top-1 accuracy was 92.79%. The accuracy achieved at this temperature was slightly higher than the others, with a very low standard deviation of 0.08%, indicating a consistent performance. Furthermore, it can be observed that varying the temperature τ from 0.1 to 8 led to minimal variations in the top-1 accuracy, with all values hovering around the 92.59% to 92.79% range. The standard deviations also were relatively low for all the experiments, suggesting that the model's performance was stable across different τ settings. This suggests that the knowledge distillation process is robust to changes in temperature τ within the explored range for the ResNet-18 model on the CIFAR-10 dataset.

E. Comparison With SOTA

In order to provide a quantitative assessment of the efficacy of DG2PF, we conducted a comprehensive set of experiments on two widely used benchmark datasets, namely CIFAR-10 [61] and ImageNet [62]. We compared our proposed algorithm with various state-of-the-art techniques to demonstrate its effective performance in network pruning. Throughout our experiments, we set the number of pruning epochs, denoted as s_e, to 15, while s_{sim} to 10%, the distillation factor α to 90% and the temperature τ to 0.5. The results for the two datasets are shown in Tables VI and VII. The tables show the baseline top-1 accuracy (acc@1) for both the unpruned models and the pruned ones, sided with the difference between the two. It is crucial to note a few disparities when comparing pruning methods. While we focused on keeping uniformity in our implementations,

TABLE VI
COMPARISON WITH THE STATE-OF-THE-ART ON THE CIFAR-10 DATASET. THE COMPRESSION RATE IS SHOWN ALONGSIDE SPARSITY PERCENTAGES

Setting					acc@1 (%)		
Model	Sparsity	Method	Params	Flops	Baseline	Pruned	Difference
VGG-16	95% (20×)	Iterative Pruning* [65]	6.9 M	x0.05	-	81.46	-
		Gradual Pruning* [27]			-	90.56	-
		One-Cycle Pruning [63]			-	90.67	-
		SNIP [64]			93.24	92.91	−0.33
		DPF [52]			93.74	93.87	+0.13
		DG2PF (ours)			93.45	93.68	**+0.23**
ResNet-18	95% (20×)	Iterative Pruning* [65]	0.57 M	x0.05	-	87.54	-
		Gradual Pruning* [27]			-	92.04	-
		One-Cycle Pruning [63]			-	92.76	-
		DG2PF (ours)			92.59	92.90	+0.31
ResNet-50	95% (20×)	GraNet [66]	1.28 M	x0.05	94.75	94.44	−0.31
		Opt [67]			94.75	94.56	−0.19
		DG2PF (ours)			92.79	93.68	**+0.89**

Note: Models marked with * are obtained from [63] reimplementing the original methods. Bold values indicate better result in a column.

TABLE VII
COMPARISON WITH THE STATE-OF-THE-ART ON THE IMAGENET DATASET. THE COMPRESSION RATE IS SHOWN ALONGSIDE SPARSITY PERCENTAGES

Setting					acc@1 (%)		
Model	Sparsity	Method	Params	Flops	Baseline	Pruned	Difference
ResNet-18	90% (10×)	One-shot Pruning* [37]	1.15 M	0.10x	69.70	63.50	−6.20
		Gradual Pruning* [37]			69.70	63.60	−6.10
		Cyclical Pruning [28]			69.70	64.90	−4.80
		DG2PF (ours)			69.70	65.22	**−4.48**
ResNet-50	90% (10×)	SWD [69]	2.56 M	0.10x	-	73.10	-
		MLPrune [71]	2.56 M	0.10x	77.01	60.98	−16.03
		PBW [72]	2.56 M	0.10x	77.01	69.44	−7.57
		RIGL [49]	2.56 M	0.13x	77.01	72.0	−5.01
		Gradual Pruning* [37]	2.56 M	0.10x	76.16	71.90	−4.26
		One-shot Pruning* [37]	2.56 M	0.10x	76.16	72.80	−3.36
		GMP [73]†	2.56 M	0.10x	77.01	73.91	−3.1
		DNM [74]†	2.56 M	0.10x	77.01	74.0	−3.01
		Cyclical Pruning [28]	2.56 M	0.10x	76.16	73.30	−2.86
		STR [70]	2.49 M	0.09x	77.01	74.31	−2.7
		GraNet [66]	2.56 M	0.16x	76.8	74.2	−2.6
		DG2PF (ours)	2.56 M	0.10x	76.13	73.62	**−2.51**
MobileNet V2	70% (3.33×)	Gradual Pruning* [37]	1.03 M	0.33x	71.70	61.30	−10.40
		One-shot Pruning* [37]			71.70	62.70	−9.00
		Cyclical Pruning [28]			71.70	64.40	−7.30
		DG2PF (ours)			71.71	65.59	**−6.12**

Note: Models marked with * are obtained from [28] reimplementing the original methods. Method with superscript † indicates that the data reported is obtained from reimplementation by [70]. Bold values indicate better result in a column.

the baseline accuracy among models with the same architecture may differ. This variation stems from different pretrained weights adopted by each study. As a significant number of these weights are inaccessible to the public, the replication of the exact initializations is unfeasible. Based on these assumptions, our evaluation criteria do not involve directly comparing the best scores between models with the same architecture but possibly different weights. Instead, we gave prominence to the relative accuracy difference between the pruned and unpruned versions of the same model, offering a more insightful measure of a method's efficacy.

1) CIFAR-10: We compared VGG-16 [2], ResNet-18, and ResNet-50 [3] architectures for CIFAR-10 [61] classification and evaluated our DG2PF algorithm against One-Cycle Pruning [63], SNIP [64], Iterative Pruning [65], Gradual Pruning [27], and DPF [52]. The performance comparisons are presented in Table VI. The results of our experiments showed that DG2PF outperformed all the benchmarked models, achieving the highest top-1 accuracy on all the tested architectures given the same sparsity levels. Specifically, on VGG-16, our algorithm achieved an improvement of 0.23% top-1 accuracy over the baseline and 0.1% over [52]. ResNet-18 and ResNet-50 both overcome the baseline by 0.31% and 0.89%, respectively. To the best of our knowledge and also according to a recent review [65], our work is the first one which deals with the ResNet-50 architecture in this specific application area.

2) ImageNet: As part of our research, we tested several deep learning architectures for ImageNet [62] classification, including ResNet-18, ResNet-50 [3], and MobileNet v2 [68]. We evaluated the effectiveness of our DG2PF algorithm against state-of-the-art pruning techniques, such as One-Shot Pruning [37], Gradual Pruning [27], Cyclical Pruning [28], and SWD [69]. The performance comparison is shown in Table VII. We can see that DG2PF outperforms the competitors on all the benchmarked models, yielding an improvement of 0.32% top-1 accuracy on ResNet-18 and ResNet-50, and 1.19% on MobileNet V2 against the previous best scores of [28]. The results show that DG2PF performed well on this more extensive dataset, achieving better accuracy than existing methods.

V. Conclusion, Limitations, and Future Works

We have introduced DG2PF, a novel and comprehensive algorithm that gradually prunes pretrained NNs using magnitude-based unstructured pruning techniques and knowledge distillation. The method has been designed to minimize performance loss due to compression. Based on a well-known pruning function, a specified proportion of weights from a pretrained NN is selectively removed to minimize memory and storage requirements. A novel simulated pruning strategy with the advantages of weight recovery and without the disadvantages of unstable convergence has also been presented. The combination of those techniques is used in the DGP phase of the algorithm. Then, the PF phase further supports the performance recovery due to the pruning. The algorithm's effectiveness has been rigorously evaluated on publicly available benchmark datasets and models, demonstrating significant improvements in memory usage and computational efficiency while maintaining high accuracy. Consequently, this method provides a promising avenue for optimizing pruned pretrained NNs with potential applications in various domains.

For future works, there are several areas to explore. One avenue is to investigate different pruning functions to determine their effectiveness in reducing memory and storage requirements while maintaining accuracy. The simulated pruning strategy can also be enhanced to achieve even better weight recovery and convergence properties. Additionally, exploring domain-specific applications and scaling up the algorithm to larger models would further validate its effectiveness. This study supports the following assumption: weights closer to zero have less impact on the final prediction in comparison to larger values for magnitude-based pruning methods [27], [28], [45]. Despite the actual results shown in this method and the related work, it is crucial to recognize the limitations of this assumption. For instance, research indicates that Transformer-based networks typically achieve a lower level of sparsity using this class of pruning algorithms [75], [76], [77]. Acknowledged that our method can indeed be adapted to different activation functions and network architectures, the correct adjustments might be essential to accommodate the specific attributes of these networks in future work findings. Last, integrating the algorithm with other optimization techniques, such as quantization and network architecture search, could yield even better results. Overall, the DG2PF algorithm presents a comprehensive solution for optimizing pruned pretrained NNs, and future research can further improve its performance and applicability in various domains.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, San Diego, CA, USA: Curran Associates, 2012, pp. 1–9. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015, pp. 1–14. [Online]. Available: https://arxiv.org/abs/1409.1556

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2016, pp. 770–778. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.90

[4] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2016, pp. 779–788. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.91

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. San Diego, CA, USA: Curran Associates, 2015, pp. 1–9. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

[7] E. Mohamed, A. M. Shaker, H. Rashed, A. E. Sallab, and M. M. Hadhoud, "Insta-yolo: Real-time instance segmentation," 2021. [Online]. Available: https://arxiv.org/abs/2102.06777

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2980–2988. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

[9] B. Wu et al., "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Piscataway, NJ, USA: IEEE, 2019, pp. 10734–10742. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Wu_FBNet_Hardware-Aware_Efficient_ConvNet_Design_via_Differentiable_Neural_Architecture_Search_CVPR_2019_paper.html

[10] Y. Guo et al., "Breaking the curse of space explosion: Towards efficient NAS with curriculum search," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 3822–3831. [Online]. Available: https://proceedings.mlr.press/v119/guo20b.html

[11] Y. Chen et al., "Contrastive neural architecture search with neural architecture comparators," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9502–9511. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Chen_Contrastive_Neural_Architecture_Search_With_Neural_Architecture_Comparators_CVPR_2021_paper.html

[12] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2019. [Online]. Available: https://arxiv.org/abs/1908.09791

[13] Y. Guo et al., "Towards accurate and compact architectures via neural architecture transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6501–6516, Oct. 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9447923?casa_token=y_OTboxHRbEAAAAA:RO3n414-jNsa_jXsoeEpzE_tr4Wlf7r-SvLkEZ2FZiLA_pHzgiT0qtdU8PejmsUIy9FMOzO1Qw

[14] S. Niu et al., "Disturbance-immune weight sharing for neural architecture search," *Neural Netw.*, vol. 144, no. 1, pp. 553–564, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360802100352X

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," *Int. Conf. Learn. Representations*

(ICLR), 2017, pp. 1–13. [Online]. Available: https://openreview.net/forum?id=rJqFGTslg

[17] S. Gao, F. Huang, W. Cai, and H. Huang, "Network pruning via performance maximization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9270–9280. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Gao_Network_Pruning_via_Performance_Maximization_CVPR_2021_paper.html

[18] X. Ding et al., "ResRep: Lossless CNN pruning via decoupling remembering and forgetting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 4510–4520. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Ding_ResRep_Lossless_CNN_Pruning_via_Decoupling_Remembering_and_Forgetting_ICCV_2021_paper.html?ref=https://githubhelp.com

[19] J. Liu et al., "Discrimination-aware network pruning for deep model compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4035–4051, Mar. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9384353

[20] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, no. 1, pp. 370–403, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221010894

[21] X. Ma et al., "Non-structured DNN weight pruning—Is it beneficial in any platform?" *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 4930–4944, Mar. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9381660

[22] S. Huang, C. Pearson, R. Nagi, J. Xiong, D. Chen, and W.-m. Hwu, "Accelerating sparse deep neural networks on FPGAS," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Waltham, MA, USA, 2019, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/8916419

[23] J. Li and A. Louri, "AdaPrune: An accelerator-aware pruning technique for sustainable CNN accelerators," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 1, pp. 47–60, Jan./Mar. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9359522

[24] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605. [Online]. Available: https://proceedings.neurips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html

[25] W. Lei, H. Chen, and Y. Wu, "Compressing deep convolutional networks using K-means based on weights distribution," in *Proc. 2nd Int. Conf. Intell. Inf. Process.*, New York, NY, USA: ACM, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3144789.3144803

[26] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016. [Online]. Available: https://arxiv.org/abs/1611.06440

[27] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," Oct. 2017. [Online]. Available: https://arxiv.org/abs/1710.01878

[28] S. Srinivas, A. Kuzmin, M. Nagel, M. van Baalen, A. Skliar, and T. Blankevoort, "Cyclical pruning for sparse neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2022, pp. 2761–2770. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPRW56347.2022.00312

[29] Y. Li, K. Adamczewski, W. Li, S. Gu, R. Timofte, and L. V. Gool, "Revisiting random channel pruning for neural network compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2022, pp. 191–201. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.00029

[30] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014. [Online]. Available: https://arxiv.org/abs/1412.6550

[31] M. Kang and S. Kang, "Data-free knowledge distillation in neural networks for regression," *Expert Syst. Appl.*, vol. 175, no. 1, 2021, Art. no. 114813. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421002542

[32] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4388–4403, Aug. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9381661

[33] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," *Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 1398–1406. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Channel_Pruning_for_ICCV_2017_paper.pdf

[34] J. Choquette and W. Gandhi, "NVIDIA A100 GPU: Performance & innovation for GPU computing," in *Proc. IEEE Hot Chips 32 Symp. (HCS)*, Palo Alto, CA, USA: IEEE Computer Society, 2020, pp. 1–43. [Online]. Available: https://ieeexplore.ieee.org/document/9220622

[35] A. Ignatov et al., "AI benchmark: Running deep neural networks on android smartphones," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, 2018, pp. 288–314. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-11021-5_19

[36] J. Liu et al., "Discrimination-aware network pruning for deep model compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4035–4051, Mar. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9384353

[37] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.,* Cambridge, MA, USA: MIT Press, 2015, vol. 1, pp. 1135–1143. [Online]. Available: https://dl.acm.org/doi/10.5555/2969239.2969366

[38] Y. Bai, H. Wang, X. Ma, Y. Zhang, Z. Tao, and Y. Fu, "Parameter-efficient masking networks," *Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 10,217–10,229. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/427048354ac2db22d43149c51346bafd-Abstract-Conference.html

[39] Y. Chen, Z. Ma, W. Fang, X. Zheng, Z. Yu, and Y. Tian, "A unified framework for soft threshold pruning," 2023. [Online]. Available: https://arxiv.org/abs/2302.13019

[40] A. Kusupati et al., "Soft threshold weight reparameterization for learnable sparsity," in *Proc. Int. Conf. Mach. Learn. (UCML)*, Jul. 2020. [Online]. Available: https://dl.acm.org/doi/10.5555/3524938.3525452

[41] K. Azarian, Y. Bhalgat, J. Lee, and T. Blankevoort, "Learned threshold pruning," 2020. [Online]. Available: https://arxiv.org/abs/2003.00075

[42] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 164–171. [Online]. Available: https://proceedings.neurips.cc/paper/1992/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html

[43] L. Theis, I. Korshunova, A. Tejani, and F. Huszár, "Faster gaze prediction with dense networks and fisher pruning," 2018. [Online]. Available: https://arxiv.org/abs/1801.05787

[44] S. P. Singh and D. Alistarh, "WoodFisher: Efficient second-order approximations for model compression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1–12. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/d1ff1ec86b62cd5f3903ff19c3a326b2-Abstract.html

[45] C. Laurent, C. Ballas, T. George, P. Vincent, and N. Ballas, "Revisiting loss modelling for unstructured pruning," 2021. [Online]. Available: https://openreview.net/forum?id=jpm1AfJucwt

[46] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, I. Guyon et al., Eds., San Diego, CA, USA: Curran Associates, 2017, pp. 3288–3298. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/69d1fc78dbda242c43ad6590368912d4-Abstract.html

[47] B. Dai, C. Zhu, B. Guo, and D. Wipf, "Compressing neural networks using the variational information bottleneck," in *Proc. Int. Conf. Mach. Learn. (ICML)*, PMLR, 2018, pp. 1135–1144. [Online]. Available: http://proceedings.mlr.press/v80/dai18d/dai18d.pdf

[48] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1902.09574

[49] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, JMLR.org, 2020, pp. 2943–2952. [Online]. Available: http://proceedings.mlr.press/v119/evci20a/evci20a.pdf

[50] S. Jayakumar, R. Pascanu, J. Rae, S. Osindero, and E. Elsen, "Top-KAST: Top-K always sparse training," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. San Diego, CA, USA: Curran Associates, 2020, pp. 20, 744–20, 754. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/ee76626ee11ada502d5dbf1fb5aae4d2-Paper.pdf

[51] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1387–1395. [Online]. Available: https://dl.acm.org/doi/10.5555/3157096.3157251

[52] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, "Dynamic model pruning with feedback," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=SJem8lSFwB

[53] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: https://arxiv.org/abs/1503.02531

[54] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), 2017, pp. 7130–7138. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Yim_A_Gift_From_CVPR_2017_paper.pdf

[55] J. Park and A. No, "Prune your model before distill it," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., Cham, Switzerland: Springer Nature Switzerland, 2022, pp. 120–136. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-20083-0_8

[56] N. Aghli and E. Ribeiro, "Combining weight pruning and knowledge distillation for CNN compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2021, pp. 3191–3198. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021W/EVW/papers/Aghli_Combining_Weight_Pruning_and_Knowledge_Distillation_for_CNN_Compression_CVPRW_2021_paper.pdf

[57] Z. Tang et al., "Automatic sparse connectivity learning for neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7350–7364, Oct. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9690593

[58] R. Meyer and A. Wong, "A fair loss function for network pruning," in *Proc. Workshop Trustworthy Socially Responsible Mach. Learn. (TSRML)*, 2022, pp. 1–18. [Online]. Available: https://arxiv.org/abs/2211.10285

[59] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. Yun, "Comparing Kullback-Leibler divergence and mean squared error loss in knowledge distillation," 2021. [Online]. Available: https://arxiv.org/abs/2105.08919

[60] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," Nov. 2017. [Online]. Available: https://arxiv.org/abs/1711.05101

[61] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009. [Online]. Available: https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf

[62] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.* (*IJCV*), vol. 115, no. 3, pp. 211–252, 2015. [Online]. Available: https://link.springer.com/article/10.1007/s11263-015-0816-y

[63] N. Hubens, M. Mancas, B. Gosselin, M. Preda, and T. Zaharia, "One-cycle pruning: Pruning ConvNets with tight training budget," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2022, pp. 4128–4132. [Online]. Available: https://ieeexplore.ieee.org/document/9897980

[64] N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: Single-shot network pruning based on connection sensitivity," 2018. [Online]. Available: https://arxiv.org/abs/1810.02340

[65] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" in *Proc. Mach. Learn. Syst.*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 129–146. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2020/file/6c44dc73014d66ba49b28d483a8f8b0d-Paper.pdf

[66] S. Liu et al., "Sparse training via boosting pruning plasticity with neuroregeneration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 9908–9922. [Online]. Available: https://openreview.net/pdf?id=MNVjrDpu6Yo

[67] Y. Zhang, M. Lin, M. Chen, F. Chao, and R. Ji, "OptG: Optimizing gradient-driven criteria in network sparsity," 2022. [Online]. Available: https://arxiv.org/abs/2201.12826

[68] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4510–4520. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[69] H. Tessier, V. Gripon, M. Léonardon, M. Arzel, T. Hannagan, and D. Bertrand, "Rethinking weight decay for efficient neural network pruning," *J. Imag.*, vol. 8, no. 3, pp. 1–23, 2022. [Online]. Available: https://www.mdpi.com/2313-433X/8/3/64

[70] A. Kusupati et al., "Soft threshold weight reparameterization for learnable sparsity," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, JMLR.org, 2020, pp. 5544–5555. [Online]. Available: http://proceedings.mlr.press/v119/kusupati20a/kusupati20a.pdf

[71] W. Zeng and R. Urtasun, "MLPrune: Multi-layer pruning for automated neural network compression," 2019. [Online]. Available: https://openreview.net/forum?id=r1g5b2RcKm

[72] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015. [Online]. Available: https://arxiv.org/abs/1510.00149

[73] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017. [Online]. Available: https://arxiv.org/abs/1710.01878

[74] M. Wortsman, A. Farhadi, and M. Rastegari, "Discovering neural wirings," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dé-Buc, E. Fox, and R. Garnett, Eds., vol. 32. San Diego, CA, USA: Curran Associates, 2019, pp. 1–11. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/d010396ca8abf6ead8cacc2c2f2f26c7-Paper.pdf

[75] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, "Chasing sparsity in vision transformers: An end-to-end exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. San Diego, CA, USA: Curran Associates, 2021, pp. 19, 974–19, 988. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/a61f27ab2165df0e18cc9433bd7f27c5-Paper.pdf

[76] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, "Width & depth pruning for vision transformers," *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 3, pp. 3143–3151, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20222

[77] L. Yu and W. Xiang, "X-Pruner: Explainable pruning for vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 24355–24363. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023/html/Yu_X-Pruner_eXplainable_Pruning_for_Vision_Transformers_CVPR_2023_paper.html