# KNAPSACK PRUNING WITH INNER DISTILLATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Neural network pruning reduces the computational cost of an over-parameterized network to improve its efficiency. Popular methods vary from $\ell_1$-norm sparsification to Neural Architecture Search (NAS). In this work, we propose a novel pruning method that optimizes the final accuracy of the pruned network and distills knowledge from the over-parameterized parent network's inner layers. To enable this approach, we formulate the network pruning as a Knapsack Problem which optimizes the trade-off between the importance of neurons and their associated computational cost. Then we prune the network channels while maintaining the high-level structure of the network. The pruned network is fine-tuned under the supervision of the parent network using its inner network knowledge, a technique we refer to as the *Inner Knowledge Distillation*. Our method leads to state-of-the-art pruning results on ImageNet, CIFAR-10 and CIFAR-100 using ResNet backbones. To prune complex network structures such as convolutions with skip-links and depth-wise convolutions, we propose a block grouping approach to cope with these structures. Through this we produce compact architectures with the same FLOPs as EfficientNet-B0 and MobileNetV3 but with higher accuracy, by $1\%$ and $0.3\%$ respectively on ImageNet, and faster runtime on GPU.

## 1 INTRODUCTION

Deep and wide networks such as VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2015) and EfficientNet (Tan & Le, 2019) achieve high classification accuracy on challenging benchmarks such as ImageNet (Deng et al., 2009). While these architectures perform well, in many scenarios it is desired to reduce their computational cost and model size. One approach to achieve this goal is via network pruning which has been a topic of research for decades (Lecun et al., 1989). Network pruning is a way to identify and remove the insignificant parameters of a network. These are the ones with little effect on accuracy.

Previous pruning methods show promising results. However, they suffer from two key shortcomings. The first is the heuristic expression of the problem, often relying on a coarse approximation of the contribution of each weight to the final accuracy, neglecting its direct trade off with its computational cost, or through a non-convex optimization problem that can lead to sub-optimal solution. For example, (Molchanov et al., 2019; Han et al., 2015; Li et al., 2017; Yang et al., 2018; Yu et al., 2018; Hu et al., 2016) measure the post-factum empirical influence of several pruning options in order to choose the best one, where only the pruning criterion is used to decide if to keep the weight. The second is not leveraging the expressive power of the parent network. Knowledge Distillation (KD) (Hinton et al., 2015) from the unpruned network could improve performance as shown by (Dong & Yang, 2019) who used KD on the network outputs to fine-tune the child network. Their approach, however, does not leverage to the full extent the fact that the inner structures of the unpruned and pruned networks are highly isomorphic.

In this paper we present a novel pruning approach that optimizes explicitly on the trade-off between accuracy and computational cost. Our first key idea is to formulate the pruning as a Knapsack Problem which enables the trade-off optimization. The second key idea is to introduce an *Inner Knowledge Distillation* (IKD) mechanism between the inner layers of the pruned and unpruned network. The IKD guides the child network to reproduce the inner layer's mapping patterns of the unpruned parent network as much as possible, leading to higher accuracy after fine-tuning.

The integration of the above two key ideas allows us to develop a novel method with strong empirical performance. Our method is one-shot, fast and does not require iterative re-training during pruning. The Knapsack formulation we suggest enables the pruning of non-sequential convolutions such as skip-connections and Squeeze-and-Excitation modules which are common in modern architectures,

e.g., ResNet and EfficientNet (Tan & Le, 2019). We show that our method leads to state-of-the-art results on ImageNet, CIFAR-10 and CIFAR-100 when using ResNets and EfficientNets as backbones.

The structure of the paper is as follows: In Section 2, we briefly review previous works on pruning and knowledge distillation. In Section 3, we describe the technical aspects of our method to prune sequential convolutions or convolutions that are not connected to a skip-connection. In Section 4, we extend our method to more complicated architectures, that include skip-connections, dilated convolutions or Squeeze-and-Excitation modules which enforce constraints on the convolutional channels to be pruned. In Section 5, we describe our fine-tuning method with IKD. Finally, in Section 6, we present the results of our pruning method on different benchmarks and backbones.

## 2 Related Works

In this section, we briefly review previous works on pruning and knowledge distillation that closely relate to our work.

Network Pruning Network pruning dates back to (Lecun et al., 1989) where the importance of a neuron is estimated by the diagonal elements of the Hessian matrix of the network's loss function. For modern neural networks estimating the Hessian matrix is prohibitive due to the high dimensionality. Therefore, inspired by the success of compressed sensing techniques (Donoho, 2006), many $\ell_1$-norm sparsification methods and sparse proximal projection methods have been introduced to prune over-parameterized networks (Liu et al., 2015; Ding et al., 2019b; Liu et al., 2017). These methods require iterative pruning during training which makes them inapplicable to pre-trained networks.

Methods that perform post-training pruning over pre-trained neural networks are under active research (Han et al., 2015; Li et al., 2017; Yang et al., 2018; Yu et al., 2018; Hu et al., 2016). Their key idea is estimating the importance of a neuron via heuristics. A comprehensive comparison of pruning heuristics is presented in (Molchanov et al., 2017), including Minimum $\ell_2$ Weight, Activation, Mutual Information, Taylor Expansion, and Average Percentage of Zeros. They show that the best criterion is the Taylor Expansion which approximates the change in the loss function induced by the pruning.

More recently, (Molchanov et al., 2019) demonstrated the high correlation between the importance approximation to a reliable estimate of the true importance of the neurons. However, their decision of removing $N$ neurons with the smallest importance scores is rather heuristic and does not account for the induced change of FLOPs.

Knowledge Distillation Knowledge distillation refers to training a student network using a teacher network by distilling information from the teacher to the student. (Hinton et al., 2015) uses a penalty term consisting of the cross entropy between the output logits of the teacher and that of the student in the loss function. A few methods use knowledge distillation inside the network. For example, (Li et al., 2019; Wang et al., 2018) consider the $\ell_2$ distance between the teacher and the student feature maps as part of the loss. When the dimensions of the feature maps of the two networks differ, a popular method is to penalize the distance between the embeddings of the features maps in a subspace of lower dimension. For instance, (Crowley et al., 2018) computes the $\ell_2$ distance between the squared sum of the teacher and the student feature maps while (Tung & Mori, 2019) penalizes the distance between the activation correlation matrices. A distillation at the level of the feature maps has been already studied by previous works such as (Romero et al., 2015; Heo et al., 2019), but the internal feature maps on which the distillation is performed are chosen arbitrarily.

Knapsack Problem The knapsack problem is extensively used in a wide variety of fields including financial trading (Markowitz & Manne, 1957), cryptography (Odlyzko, 1990) and resource distribution (Vanderster et al., 2009). Recent works utilize deep neural networks for efficient and accurate optimization for solving the knapsack problem (Gu & Hao, 2018; Martini, 2019) To the best of our knowledge, this work is the first to utilize a Knapsack Problem to prune deep neural networks.

## 3 Methodology to Prune Sequential Convolutions

In this section, we present our method for pruning sequential convolutions. This allows us to prune networks such as VGG as well as all the convolutions inside ResNet that are not preceding a skip-connection. Generalization to non-sequential operations such as skip-connections or integration of operations, is presented in Section 4.

### 3.1 Knapsack Problem and Pruning

Suppose we have a knapsack with a capacity $C$ and a collection of $n$ items $\mathcal{I}$ where every item $o_i \in \mathcal{I}$ has a weight $f_i$ and a value $v_i$. The Knapsack Problem aims to fill the knapsack with maximal value,

considering the weight capacity $C$. That is

$$\max_{\boldsymbol{b}} \sum_i v_i b_i \quad \text{s.t} \sum_i f_i b_i \leq C,\ b_i \in \{0,1\} \quad 1 \leq i \leq n \tag{1}$$

where the indicator variable $b_i$ equals 1 if $o_i$ is selected and 0 otherwise.

The above formulation is an integer programming problem which is NP-hard. If the weights $f_i$ are integers, the problem has an exact solution that can be found with a Dynamic Programming algorithm in a $O\left(n \max_i f_i\right)$ time complexity. An approximate solution of the problem can also be found with a greedy approximation algorithm (Dantzig, 1957) in $O(n\log(n))$ time complexity. The method relaxes the original problem by replacing the constraint $b_i \in \{0,1\}$ with $0 \leq b_i \leq 1$ . Then the approximated solution can be derived in a closed form.

We formulate the network pruning task as a approximate Knapsack problem. Given a network $\mathcal{N}$ with convolutional layers $\mathcal{C}_l, 1 \leq l \leq L$, we seek to prune its output channels with the least impact on the classification accuracy under a target FLOPs budget $C$. Denote by $\mathcal{P}_\mathcal{N}$ the space of pruned versions of $\mathcal{N}$ and by $\mathrm{Acc}$ the accuracy on a validation set $\mathcal{X}$. We formulate the problem as follows:

$$\max_{\mathcal{N}_{\text{pruned}}} \mathrm{Acc}(\mathcal{N}_{\text{pruned}}, \mathcal{X}) \quad \text{s.t}\, \mathcal{N}_{\text{pruned}} \in \mathcal{P}_\mathcal{N},\ \mathrm{FLOPs}(\mathcal{N}_{\text{pruned}}) \leq C \tag{2}$$

Optimizing the above problem is not straightforward as the accuracy $\mathrm{Acc}$ is not differentiable. Therefore, it is common to use an approximated formulation that minimize the cross-entropy loss to replace the $\mathrm{Acc}$.

Yet, Eq. (2) remains costly to solve, therefore we next propose an additional approximation. Instead of maximizing the accuracy (minimizing the cross-entropy loss), we minimize the change of the loss due to zeroing-out the pruned network neurons. Correspondingly, we adjust the constraint of FLOPs to constrain the accumulated FLOPs that are associated with the selected weights. The space $\mathcal{P}_\mathcal{N}$ can be represented with a binary indicator vector $\mathbf{b}$ where $b_i \in \{0,1\}$ indicates if the network's weight $w_i$ is zero or not. We denote by $I(w_i)$ the change of the loss $\mathcal{L}_{\text{CE}}(x, \mathcal{N}_{\text{pruned}})$ and by $F(w_i)$ the saving of the FLOPs when setting $b_i$ to zero. Problem (2) can be now approximated as:

$$\max_{\boldsymbol{b}} \sum_i b_i I(w_i) \quad \text{s.t} \sum_i b_i F(w_i) \leq C,\ b_i \in \{0,1\}\ \forall i \tag{3}$$

The above Eq. (3) is equivalent to the Knapsack Problem Eq. (1). We will now describe how we compute $I(w_i)$ and $F(w_i)$.

The change of loss $I(w_i)$ can be approximated by the first order Taylor Expansion of the loss function (Molchanov et al., 2017). Formally, given a function $f : \mathbb{R}^n \to \mathbb{R}$ and a vector $\boldsymbol{w} \in \mathbb{R}^n = \sum_i w_i \boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the $i$-th canonical vector of $\mathbb{R}^n$ filled with 0 everywhere except for the 1-th coordinate. Denote $\tilde{\boldsymbol{w}}^j = \sum_{i \neq j} w_i \boldsymbol{e}_i$ a copy of the vector $\boldsymbol{w}$ with the $j$-th coordinate replaced by zero. We have

$$f(\tilde{\boldsymbol{w}}^j) = f(\sum_{i \neq j} w_i \boldsymbol{e}_i) \approx f(\boldsymbol{w}) - w_j \frac{\partial f(\boldsymbol{w})}{\partial w_j}.$$

Therefore the impact on the loss of zeroing the weight $w_l^o$ of the $o$-th output channel of the $l$-th layer can be approximated by:

$$I(w_l^o) \approx -\mathbb{E}_{\mathbf{x}} \left( {w_l^o}^T \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{w})}{\partial w_l^o} \right) \tag{4}$$

where $\mathbf{x}$ is the input instances (images for example). The higher this value, the higher the impact of the weight on the total loss. Unfortunately, the above approximation may be too noisy since the

expectation of the gradient is zero at the convergence point of the loss function. In (Molchanov et al., 2017), they show that the variance of the quantity $z_l^o = {w_l^o}^T \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{w})}{\partial w_l^o}$ is usually non-zero and correlates with the stability of the local function with respect to $w_l^o$ proposing the following approximation instead:

$$I(w_l^o) \approx \mathbb{E}_{\mathbf{x}} \left| {w_l^o}^T \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{w})}{\partial w_l^o} \right|. \tag{5}$$

Empirically, we observe that using the below approximation leads to better performances:

$$I(w_l^o) \approx \mathbb{E}_{\mathbf{x}} \left( \left| {w_l^o}^T \right| \left| \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{w})}{\partial w_l^o} \right| \right). \tag{6}$$

In practice, the expectation in Eq. (4) can be approximated by averaging over a validation set.

Last, we need a formula to calculate the saving of FLOPs $F(w_i)$ after removing the network weight $w_l^o$. Up to now, we focus on the single weight. But in pruning we remove weights in groups. More particularly, we remove a group of weights that are used to compute a channel, such as a filter in a common convolutional layer. Given a convolution with $C_i^l$ input channels of size $H^l \times W^l$ and $C_o^l$ output channels with kernel size $k^l \times k^l$ and stride $s^l$, its FLOPs is $C_o^l C_i^l H^l W^l (k^l)^2/(s^l)^2$. Zeroing a group of weights related to $w_l^o$ requires removing both an output channel from layer $\mathcal{C}_l$ and an input channel from layer $\mathcal{C}_{l+1}$. Therefore, the saving of FLOPs is given by

$$F(w_l^o) = \frac{C_i^l H^l W^l (k^l)^2}{(s^l)^2} + \frac{C_o^{l+1} H^{l+1} W^{l+1} (k^{l+1})^2}{(s^{l+1})^2}. \tag{7}$$

Solving the Knapsack Problem (3) could be done via dynamic programming. The complexity of the dynamic programming is $O(nF_{\text{max}})$, and in our case, $F_{\text{max}} = \max_i F(w_l^o)$ represents the maximum FLOPs required by a convolutional channel of the network, and can be computed with Eq. (7). In practice, we can reduce the computational complexity from $O(nF_{\text{max}})$ to $O\left(nF_{\text{max}}/g\right)$, where $g$ is the Greatest Common Divisor (GCD) of the set $\{F(w_l^o)\ \forall 1 \leq l \leq L\}$. Dividing both $F(w_l^o)$ and $C$ by $g$ accelerates the convergence time without changing the solution. The total knapsack runtime is negligible in comparison to the network fine-tuning process discussed in Section 5. The details of the optimization procedure are described in Algorithm (1) in the supplementary. In addition, we can replace the FLOPS constraint by a running time constraint. In the supplementary material, we present the results of some networks trained with such a method, and show that our formulation allows to get time-pruned networks with the highest accuracy for a given inference time constraint.

## 4 Pruning Non-Sequential Convolutions

To date, most pruning methods are restricted to sequential connections as non-sequential connections are non trivial to prune. For example, pruning a filter whose output is to be added to the output of another filter is problematic, because the addition operation implies that their impact is intertwined. Even though there are several methods that prune ResNet, most of them will either avoid pruning the convolutions connected to a skip-connection, provide a network whose inherent structure is not pruned per-se but contains sparse convolutions, or use a scatter-add operation that empirically can take more time than the gain by pruning itself.

We now suggest a novel method that allows pruning of non-sequential connections as part of the proposed knapsack framework. The key idea is to group operations that together directly form a channel or a group of channels in a feature map, such as all the convolutions whose outputs are connected through a summation, a multiplication or any inherent constraint like the one in separable convolution. In this setting, the channels of every group are pruned together, and the pruned network structure is consistent with the unpruned one. This is only possible due to our knapsack formulation.

To make this more clear we take as an example a cell called *inverted residual* as shown in Figure 1 where we neglect activation functions for brevity. This cell appears in EfficientNet (Tan & Le, 2019), MNASnet (Tan et al., 2018) and MobileNet (Howard et al., 2019). This cell contains both Squeeze-and-Excitation components (Hu et al., 2018) and dilated convolutions.

There are three constraints on the inverted residual block. First, the output channels of the 'Point-wise linear projection' have to match the input of the current block because of the skip-connection. Second, the output channels of the 'Point-wise expansion' have to match the output channels of the 'Depth-wise convolution' since a Depth-wise convolution has a number of output channels that corresponds the the number of input channels. Lastly, the output channels of the 'Depth-wise convolution' have to match the output channels of the 'Squeeze-and-Excitation Expansion Convolution' because of the skip multiplication.

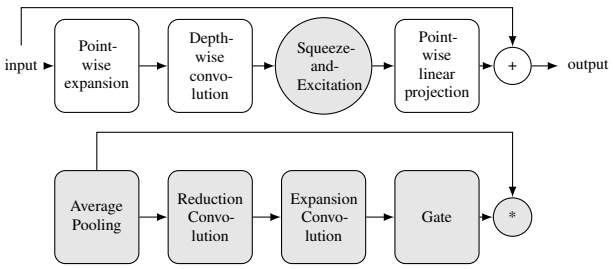



Figure 1: Inverted Residual Block with Squeeze-and-Excitation

In order to prune this cell we build three groups of convolutions. The first includes the successive 'Point-wise linear projections'. The second includes the 'Point-wise expansions', the 'Depth-wise convolutions' and 'Squeeze-and-Excitation Expansion convolutions' of the same block. The third consists of the 'Squeeze-and-Excitation Reduction Convolutions'. As mentioned above, for each of these three groups we prune their associated channels together.

To the best of our knowledge, we are the first to suggest a pruning method that applies effectively to a non-sequential architecture such as EfficientNet.

## 5 Inner Knowledge Distillation and Fine-Tuning

After we get the architecture of the pruned network, we fine-tune its weights. Here we present a method that accelerates the process of fine-tuning by reducing the number of steps. For instance, in TAS (Dong & Yang, 2019), they require 236 GPU hours to search for the pruned version of ResNet-18 using NVIDIA Tesla V100 GPUs. Our method finds the pruned network in less that 0.1 GPU hours and requires 19 GPU hours to fine-tune the network. That is 12 times faster.

A common practice in fine-tuning is to incorporate a Knowledge Distillation term (Hinton et al., 2015; Tian et al., 2020) in the loss function. This has proven to be very efficient and increases the final accuracy of a student network when using a high accuracy teacher network.

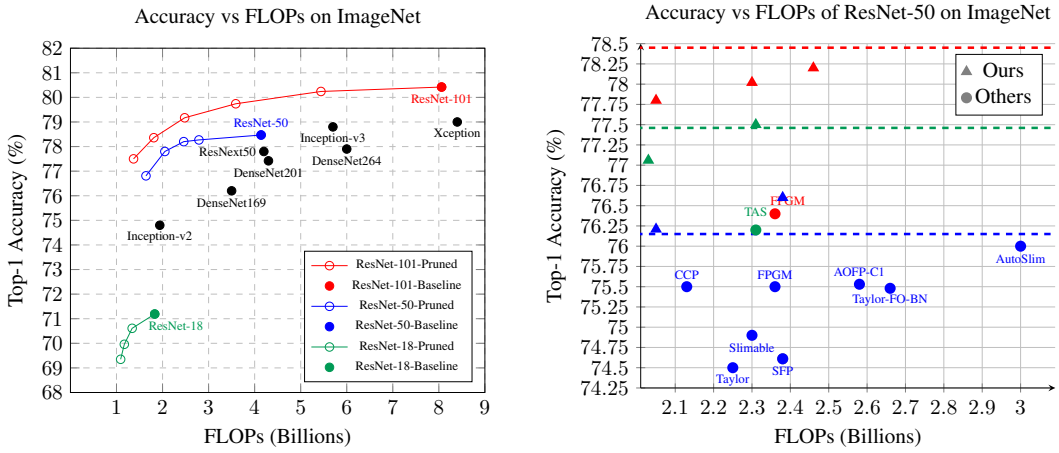
Denote by $\mathcal{N}_{\text{Teacher}}, \mathcal{N}_{\text{Student}}$, the teacher and student networks, and their respective output logits by $\mathcal{F}^t_{\text{out}}, \mathcal{F}^s_{\text{out}}$. Let $\mathrm{SM}(\cdot)$ denote the softmax operator defined by $\mathrm{SM}(\boldsymbol{y})_i = \dfrac{\exp(y_i)}{\sum_j \exp(y_j)}$. The KD enforces the output logits distributions of the teacher and student networks to be as similar as possible. This is achieved by adding Kullback–Leibler divergence in the loss function as

$$\mathcal{L}_{\text{IKD}} = \sum_{x,i} -\log\left(\mathrm{SM}\left(\mathcal{F}^s_{\text{out}}(x)\right)_i\right) \mathrm{SM}\left(\mathcal{F}^t_{\text{out}}(x)\right)_i . \tag{8}$$

We next suggest a further loss term that aims for similarity between $\mathcal{N}_{\text{Teacher}}$ and $\mathcal{N}_{\text{Student}}$, not only between their output logits but also between their internal feature maps.

A distillation at the level of the feature maps between two different networks has already been studied by previous works such as (Romero et al., 2015; Heo et al., 2019), but the internal feature maps on which the distillation is performed are chosen arbitrarily, since the teacher and student networks have different structures. In the scope of pruning, we do not have this limitation since the teacher and student networks have the exact same structure up to the number of channels in every convolution. As far as we know, we are the first to use a feature maps distillation for pruning. What allows us to perform such a distillation is the one-shot nature of our method, meaning that we choose only once the channel to be pruned, unlike iterative methods such as (Peng et al., 2019; He et al., 2019; Dong & Yang, 2019) where the choice of the weights to be pruned is constantly updated during the process.

Let $\mathcal{F}^t_l$ be the output feature map at the $l$-th layer of $\mathcal{N}_{\text{Teacher}}$ with $C^t_l$ channels. Similarly, the output feature map at the $l$-th layer of $\mathcal{N}_{\text{Student}}$ is $\mathcal{F}^s_l$ with $C^s_l$ channels. In our case, $\mathcal{N}_{\text{Teacher}}$ and $\mathcal{N}_{\text{Student}}$ have the same structure apart from their convolutional channel numbers. Hence we could transfer the knowledge inside the network at the level of the convolutional layers. Since the convolution before activation is a linear operator, we require the pruned network to reconstruct the original feature map.




(a) Comparison of deep pruned and shallower unpruned networks. Pruning ResNet-101 provides a network with less FLOPs and better accuracy than other networks.

(b) Impact of baseline. Every color is a different baseline. Red, blue and green entries are respectively from our, PyTorch and TAS baseline (dotted lines).

Figure 2: Top-1 accuracy v.s. FLOPs for pruned ResNets on ImageNet.

We call this the Inner Knowledge Distillation (IKD). Mathematically, we define the IKD loss term as

$$\mathcal{L}_{\text{IKD}} = \sum_{x} \left( \sum_{l} \left\| \mathcal{F}_l^t(x, W_t) - \boldsymbol{M}_l \mathcal{F}_l^s(x, W_s) \right\|_2^2 \right) \tag{9}$$

where $W_l$ represents the weight matrix at layer $l$ and $\boldsymbol{M}_l$ is a $(C_l^t \times C_l^s)$ matrix that aims to reconstruct the features maps $\mathcal{F}_l^t$ from $\mathcal{F}_l^s$, and is added to the list of learnable variables in the fine-tuning process. To avoid degenerate solutions, we add a weight decay regularization term to $\boldsymbol{M}_l$, that behaves like a ridge regression regularizer.

The final loss used in the fine-tuning combines the original cross-entropy loss $\mathcal{L}_{\text{CE}}$, the Knowledge Distillation loss (8) and the Inner Knowledge Distillation loss (9):

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{IKD}} \mathcal{L}_{\text{IKD}} + \lambda_{\text{KD}} \mathcal{L}_{\text{KD}} \tag{10}$$

## 6 Experiments

In this section, we present empirical results of our pruning method on three different benchmarks: ImageNet (Deng et al., 2009), CIFAR-10 and CIFAR-100 (Krizhevsky, 2009). To show robustness to the architecture, we experiment with a variety of depths of ResNet (He et al., 2015) as well as EfficientNet (Tan & Le, 2019).

The experimental protocol is as follows: We first train a full-size baseline network on the selected dataset, next we prune it using our Knapsack formulation and last we apply fine-tuning with IKD **for 50 epochs only**, even though most of the other methods fine-tune for more than 100 epochs. In both datasets, we show significant accuracy improvement compared to the other methods. **In particular, when pruning ResNet-50 on imagenet using the standard Pytorch baseline, we get an network with 1% more accuracy than other SOTA methods for same pruning ratio (see Fig 2b).**

### 6.1 ImageNet

Comparison to other pruning methods To test our method on ImageNet, we used three versions of ResNet (He et al., 2015): ResNet-18, ResNet-50, and ResNet-101.

Table 3 and Figure 2b compare our results for different pruning ratios with previous works. It can be seen that our results are consistently better than the state-of-the-art.

Comparison to common classification networks To further evaluate the benefits of our pruning approach we present in Figure 2a a comparison of the performance of our pruned networks with popular architectures: Inception (Szegedy et al., 2016), DenseNet (Huang et al., 2017), ResNext (Xie et al., 2017) and Xception (Chollet, 2017). We compare both Top-1 accuracy and computational cost (FLOPs). It can be seen that our pruned networks consistently provide higher accuracy than other networks, for a given number of FLOPs.

ABLATION STUDY Next, we present an ablation study, to assess the contribution of the various components of our approach. We took ResNet-50 as backbone and experimented with two variants: (i) With and without IKD, and (ii) our baseline training vs. PyTorch baseline. Results are presented in Table 1. For a fair comparison with regard to the impact of our baseline, we take the original implementation of FPGM (He et al., 2019) and prune our own baseline ResNet-50 instead of the original PyTorch one. Next, we prune ResNet-50 using the same baseline of $77.46\%$ top-1 accuracy as TAS(Dong & Yang, 2019). In both cases, we can see that our method provides better results, no matter the baseline we start from as can be seen in Figure 2b.

| Baseline | IKD | Prune Acc | Acc Drop | FLOPs | Prune Ratio ↓ |
|---|---|---|---|---|---|
| High | ✓ | 78.20% | 0.27% | 2.46E9 | 40.64% |
| High | ✗ | 77.12% | 1.35% | | |
| PyTorch | ✓ | 76.60% | -0.46% | 2.38E9 | 42.56% |
| PyTorch | ✗ | 76.17% | -0.03% | | |
| High | ✓ | 77.82% | 0.65% | 2.05E9 | 50.50% |
| High | ✗ | 76.70% | 1.77% | | |
| PyTorch | ✓ | 76.21% | -0.07% | 2.03E9 | 50.80% |
| PyTorch | ✗ | 75.94% | 0.21% | | |

Table 1: Ablation study of ResNet-50 on ImageNet.

*IKD:* When using IKD, we have more than $1\%$ improvement than when not using KID, both for pruning $50\%$ of ResNet-50 and pruning $41\%$. As could be expected, when using as baseline the low-accuracy network provided with PyTorch, the performance improvement by the IKD step is smaller, going from $76.17\%$ without IKD to $76.60\%$ with IKD.

*Baseline:* To measure the impact of the baseline on our method, we choose to prune ResNet-50 with the official PyTorch (Paszke et al., 2019) pre-trained weights ( $76.15\%$ accuracy on ImageNet). This is the common evaluation scheme adopted by most works. Comparing our results in Table 1 with those of previous work in Table 3 shows that our method still provides the highest accuracy.

*Knapsack:* To assess the contribution of the Knapsack formulation, we have pruned $42.6\%$ of ResNet-50 on ImageNet on the official Pytorch baseline using Molchanov's criterion only (Molchanov et al., 2019), without the Knapsack formulation and have obtained $75.26\%$ accuracy, while the addition of the Knapsack formulation (without IKD) led to $76.17\%$ accuracy, an improvement $0.91\%$. This result stands to demonstrate the importance of the Knapsack formulation, and that our results are not due to the fact that we use the Taylor Expansion criterion.

### 6.2 PRUNING THE NON-SEQUENTIAL EFFICIENTNET

As described in Section 4, our approach can be applied also to prune architectures with non-sequential convolutions and skip-connections such as EfficientNet (Tan & Le, 2019). To the best of our knowledge, this is the first attempt to prune these types of networks.

| Network | Acc | FLOPs | Speed (Im/s) |
|---|---|---|---|
| MobileNetV3 Large | 75.2% | 0.21E9 | 1730 |
| EfficientNet B0 Pruned | **75.5%** | | **2133** |
| EfficientNet B0 | 77.3% | 0.39E9 | 1230 |
| EfficientNet B1 Pruned | **78.3%** | | **1355** |
| EfficientNet B1 | 79.2% | 0.7E9 | 784 |
| EfficientNet B2 Pruned | **79.9%** | | **882** |
| EfficientNet B2 | 80.3% | 1.0E9 | 595 |
| EfficientNet B3 Pruned | **80.8%** | | **683** |
| EfficientNet B3 | 81.7% | 1.8E9 | 350 |
| EfficientNet B4 Pruned | **81.9%** | | **385** |

Table 2: Comparison of pruned and original versions of EfficientNet on GPU NVIDIA P100.

We experimented with 4 variants, comparing pruned EfficientNet B$\{n\}$ with EfficientNet B$\{n-1\}$, where $n \in \{1, 2, 3, 4\}$. For a fair comparison with the unpruned baselines, we followed the published EfficientNet training protocol without IKD. Results are presented in Table 2. It can be observed that the pruned networks achieve higher accuracy than the baselines with the same number of FLOPs. An interesting observation is that despite having the same theoretical computational complexity, the pruned networks run faster than the unpruned ones. Furthermore, our pruned version of EfficientNet B0 led to a network with the same amount of FLOPs as MobileNetV3-large (Howard et al., 2019) and a better accuracy.

### 6.3 CIFAR

For the CIFARs datasets, we train ResNet-56 on CIFAR-10 and CIFAR-100 according to the same protocol used for ImageNet while changing the number of epochs to 300. Our top-1 accuracy baseline is $94.2\%$ for CIFAR-10 and $73.55\%$ for CIFAR-100. Results and comparisons to other works can be seen on the left of Table 4.

| Model | Method | Top-1 | | Top-5 | | FLOPs | Prune Ratio |
|---|---|---|---|---|---|---|---|
| | | Prune Acc | Acc Drop | Prune Acc | Acc Drop | | |
| ResNet-18 | LCCL (Dong et al., 2017) | 66.33% | 3.65% | 86.94% | 2.29% | 1.19E9 | 34.6% |
| | SFP (He et al., 2018a) | 67.10% | 3.18% | 87.78% | 1.85% | 1.06E9 | 41.8% |
| | FPGM (He et al., 2019) | 68.41% | 1.87% | 88.48% | 1.15% | 1.06E9 | 41.8% |
| | TAS (Dong & Yang, 2019) | 69.15% | 1.50% | 89.19% | 0.68% | 1.21E9 | 33.3% |
| | Ours | 69.96% | 1.23% | 89.60% | 0.59% | 1.17E9 | 35.77% |
| | Ours | 69.35% | 1.84% | 89.23% | 0.96% | 1.09E9 | 40.01% |
| ResNet-50 | SFP (He et al., 2018a) | 74.61% | 1.54% | 92.06% | 0.81% | 2.38E9 | 41.8% |
| | CP (He et al., 2017) | - | - | 90.80% | 1.40% | 2.04E9 | 50.0% |
| | Taylor (Liu & Liu, 2018) | 74.50% | 1.68% | - | - | 2.25E9 | 44.9% |
| | AutoSlim (Yu & Huang, 2019) | 76.00% | - | - | - | 3.00E9 | 26.6% |
| | FPGM (He et al., 2019) | 75.50% | 0.65% | 92.63% | 0.21% | 2.36E9 | 42.2% |
| | SSS (Huang & Wang, 2018) | 71.82% | 4.30% | 90.79% | 2.07% | 2.33E9 | 43.4% |
| | Taylor-FO-BN (Molchanov et al., 2019) | 75.48% | 0.70% | - | - | 2.66E9 | 35.5% |
| | Slimable (Yu et al., 2019) | 74.90% | 1.20% | - | - | 2.30E9 | 44.0% |
| | CCP (Peng et al., 2019) | 75.50% | 0.65% | 92.62% | 0.25% | 2.13E9 | 48.8% |
| | AOFP-C1 (Ding et al., 2019a) | 75.53% | -0.29% | 92.69% | -0.13% | 2.58E9 | 32.88% |
| | TAS (Dong & Yang, 2019) | 76.20% | 1.26% | 93.07% | 0.48% | 2.31E9 | 43.5% |
| | Ours | 78.20% | 0.27% | 93.98% | -0.10% | 2.46E9 | 40.64% |
| | Ours | 78.02% | 0.45% | 93.88% | 0.00% | 2.30E9 | 44.47% |
| | Ours | 77.80% | 0.67% | 93.78% | 0.10% | 2.05E9 | 50.21% |
| ResNet-101 | Taylor-FO-BN (Molchanov et al., 2019) | 75.38% | - | - | - | 2.47E9 | 69.3% |
| | FPGM (He et al., 2019) | 77.32% | 0.05% | 93.56% | 0.00% | 4.51E9 | 42.2% |
| | RSNLIA (Ye et al., 2018) | 75.27% | 2.10% | - | - | 4.13E9 | 47.0% |
| | AOFP-D2 (Ding et al., 2019a) | 76.40% | 0.23% | 93.07% | 0.22% | 3.77E9 | 50.19% |
| | Ours | 79.17% | 1.25% | 94.54% | 0.63% | 2.48E9 | 69.21% |
| | Ours | 78.36% | 2.06% | 94.27% | 0.90% | 1.81E9 | 77.50% |
| | Ours | 77.56% | 2.86% | 93.68% | 1.49% | 1.37E9 | 83.00% |

Table 3: Comparison of different pruning algorithms for different ResNet backbones on ImageNet.

| Method | CIFAR-10 | | | CIFAR-100 | | |
|---|---|---|---|---|---|---|
| | Prune Acc | Acc Drop | FLOPs | Prune Acc | Acc Drop | FLOPs |
| PFEC (Li et al., 2017) | 93.06% | -0.02% | 9.09E7 (27.6%) | – | – | – |
| LCCL (Dong et al., 2017) | 92.81% | 1.54% | 7.81E7 (37.9%) | 68.37% | 2.96% | 7.63E7 (39.3%) |
| AMC (He et al., 2018b) | 91.90% | 0.90% | 6.29E7 (50.0%) | – | – | – |
| SFP (He et al., 2018a) | 93.35% | 0.56% | 5.94E7 (52.6%) | 68.79% | 2.61% | 5.94E7 (52.6%) |
| FPGM (He et al., 2019) | 93.49% | 0.42% | 5.94E7 (52.6%) | 69.66% | 1.75% | 5.94E7 (52.6%) |
| CCP (Peng et al., 2019) | 93.69% | -0.19% | 6.61E7 (47.0%) | - | - | - |
| TAS (Dong & Yang, 2019) | 93.69% | 0.77% | 5.95E7 (52.7%) | 72.25% | 0.93% | 6.12E7 (51.3%) |
| Ours | 93.83% | 0.69% | 5.79E7 (53.8%) | 72.62% | 0.93% | 6.25E7 (50.2%) |

Table 4: Comparison of different pruning algorithms for ResNet-56 on CIFAR.

## 7 Conclusion

In this paper we have presented a new formulation and method for the pruning task, which enables us to simultaneously optimize over both accuracy and FLOPs measures, as well as distill knowledge from the unpruned network. This method has provided state-of-the-art empirical results on ImageNet and CIFAR datasets, which demonstrate the effectiveness of our proposed solution. We have observed that pruning a heavy deep network with our method can provide better accuracy than a shallower one with the same computational complexity (whether the latter was designed with a Network Architecture Search method or manually). These findings may suggest that the Network Architecture Search task should focus on finding inflated over-parametrized networks, while leaving the designing of efficient networks for the pruning and knowledge distillation methods.

## REFERENCES

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1800–1807, 2017.

Elliot J. Crowley, Gavin Gray, and Amos Storkey. Moonshine: Distilling with cheap convolutions. In *Advances in Neural Information Processing Systems*, 2018.

George Dantzig. Discrete-variable extremum problems. *Operation Research*, 5(2):266–288, April 1957. ISSN 0030-364X. doi: 10.1287/opre.5.2.266. URL https://doi.org/10.1287/opre.5.2.266.

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

Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive CNN width optimization. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 1607–1616, 2019a.

Xiaohan Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, Ji Liu, et al. Global sparse momentum sgd for pruning very deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 6379–6391, 2019b.

Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5840–5848, 2017. doi: 10.1109/CVPR.2017.205.

David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

Shenshen Gu and Tao Hao. A pointer network based deep learning algorithm for 0–1 knapsack problem. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pp. 473–477. IEEE, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1135–1143. Curran Associates, Inc., 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pp. 2234–2240. AAAI Press, 2018a. ISBN 978-0-9992411-2-7.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 1398–1406, 10 2017.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV (7)*, pp. 815–832, 2018b.

Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. *arXiv e-prints*, art. arXiv:1905.02244, May 2019.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016. URL `http://arxiv.org/abs/1607.03250`.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2261–2269, 2017.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. *ECCV*, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, 2009.

Yann Lecun, John Denker, and Sara Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pp. 598–605, 01 1989.

Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely supervised neural architecture search with knowledge distillation, 2019.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall F. Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *CVPR*, pp. 806–814. IEEE Computer Society, 2015.

Chongyang Liu and Qinrang Liu. Improvement of pruning method for convolution neural network compression. In *Proceedings of the 2018 2Nd International Conference on Deep Learning Technologies*, ICDLT '18, pp. 57–60, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6473-7.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Harry M Markowitz and Alan S Manne. On the solution of discrete programming problems. *Econometrica: journal of the Econometric Society*, pp. 84–110, 1957.

Davide Martini. Application of neural network for the knapsack problem. *online PDF*, 2019.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Andrew M Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42:75–88, 1990.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5113–5122, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Adriana Romero, Samira Ebrahimi Kahou, Polytechnique Montréal, Y. Bengio, Université De Montréal, Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *in International Conference on Learning Representations (ICLR*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2818–2826, 2016.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2018.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. *ArXiv*, abs/1907.09682, 2019.

Daniel C Vanderster, Nikitas J Dimopoulos, Rafael Parra-Hernandez, and Randall J Sobie. Resource allocation on computational grids using a utility model and the knapsack problem. *Future Generation computer systems*, 25(1):35–50, 2009.

Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pp. 2769–2775. AAAI Press, 2018. ISBN 9780999241127.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5987–5995, 2017.

Tien-Ju Yang, Andrew G. Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, pp. 289–304, 2018.

Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *International Conference on Learning Representations*, 2018.

Jiahui Yu and Thomas Huang. Network slimming by slimmable networks: Towards one-shot architecture search for channel numbers. In *arXiv e-prints*, 03 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry Davis. Nisp: Pruning networks using neuron importance score propagation. In *arXiv e-prints*, pp. 9194–9203, 06 2018. doi: 10.1109/CVPR.2018.00958.