Channel Pruning via Lookahead Search Guided Reinforcement Learning

Zi Wang Chengcheng Li
University of Tennessee, Knoxville, TN, USA
{zwang84,cli42}@vols.utk.edu

Abstract

Channel pruning has become an effective yet still challenging approach to achieve compact neural networks. It aims to prune the optimal set of filters whose removal results in minimal performance degradation of the slimmed network. Due to the prohibitively vast search space of filter combinations, existing approaches usually use various criteria to estimate the filter importance while sacrificing some precision. Here we present a new approach to optimizing the filter selection in channel pruning with lookahead search guided reinforcement learning (RL). A neural network that takes as input filter-related features is trained with RL to prune the optimal sequence of filters and maximize the performance of the remaining network. In addition, we employ Monte Carlo tree search (MCTS) to provide a lookahead search for filter selection, which increases the sample efficiency for the RL training. Experiments on MNIST, CIFAR-10, and ILSVRC-2012 validate the effectiveness of our approach compared to both traditional and automated existing channel pruning approaches.

1. Introduction

In recent years, deep convolutional neural networks (CNNs) have enjoyed tremendous successes and have been widely used in different computer vision tasks , such as object classification [38, 22, 30] and detection [16, 73, 72], image synthesis [17, 69, 42, 83], semantic segmentation [55, 11, 48], and biological image-based disease diagnosis [41, 70, 87]. Improvements over the state of the art in these areas are usually achieved by designing deeper and wider CNNs [38, 78, 80, 22, 31]. However, the computing cost in these CNNs is usually expensive, which restricts their usage on resource-constrained devices such as robotics, mobile devices, and drones [59, 74, 66, 5, 65, 1, 12, 9].

To reduce the computation and storage cost of a CNN while maintaining its performance, many network compression approaches have been proposed, such as matrix decomposition [35, 94, 81], quantization [20, 71, 34], knowledge distillation [28, 79, 8, 8, 82, 84], and network pruning [21, 68, 44, 64, 27, 57, 93, 26, 43, 86]. Among all these approaches, network pruning has become the most popular one because of its appealing properties of straightforward implementation and large pruning ratio with little performance degradation. Compared to weight pruning [21, 96], channel pruning is a more straightforward approach because it removes the entire filter/channel rather than specific weights so that it can be implemented without the need for specialized software and hardware [19]. However, most of the existing channel pruning approaches are faced with the following two issues.

1) **Filters' importance heavily relies on approximation.** As we know, the optimal solution for channel pruning is to evaluate the performance of the slimmed networks after removing every possible combination of filters that meets a pre-defined pruning ratio in a trial-and-error manner [14] and choose the best one, which is called oracle pruning [64]. However, this approach is computationally insupportable due to the prohibitively vast search space of filter combinations. Therefore, many channel pruning approaches rely on the fast estimation of filter importance while sacrificing some ranking precision. For example, [44] uses a heuristic criterion that prunes the filters with the smallest weight magnitude. [26] calculates the geometric median of the filters and removes those that exhibit functional redundancy.

2) **Filters are often pruned with a greedy selection strategy without a lookahead mechanism.** In many existing channel pruning approaches, the filters' influence on the performance is estimated solely based on the current configuration of the network, and the long-range influence of these pruned filters is omitted. A recent study [14] observes that filters' importance changes dramatically after each pruning iteration. Moreover, [24] shows that keeping the identified unimportant filters in the neural network, continuing finetuning them, and re-selecting a new set of filters to prune during each iteration results in a better performance than permanently pruning the selected filters and finetuning the remaining sub-network iteratively. These studies indicate that pruning filters based on the current configuration of a network is not always the best choice without a lookahead mechanism. A deeper search needs to be implemented to

achieve optimal filter importance calculation/estimation.

With the above concerns, we propose a novel channel pruning approach with lookahead search guided reinforcement learning (RL). Specifically, we train a neural network with RL [62, 63, 61] that learns to prune a sequence of filters and maximize the performance of the slimmed network. Given a CNN to be pruned, it takes as input a number of filter-related features extracted from the CNN and outputs the probabilities of pruning each filter. After a filter is pruned based on the output probabilities, a slimmed CNN architecture is obtained, whose features are used as the input in the next iteration. The process repeats until a requirement is reached (for example, a certain amount of FLOPs are pruned). For each intermediate CNN during the pruning process, before the filter selection, a Monte Carlo tree search (MCTS) [36, 10, 7] is executed to provide a lookahead search and generate much stronger policies for RL training. With RL and MCTS incorporated in channel pruning, the sample efficiency is significantly improved and the search space is remarkably narrowed. Therefore, the pruning can be controlled within a tolerable time compared to oracle pruning. We validate our proposed approach with various benchmark network architectures and datasets. Experiment results demonstrate that our approach achieves improvements over the current state of the art of both traditional [44, 64, 26, 51] and automated [25, 15, 91, 90] channel pruning approaches.

2. Related work

The most well-known pruning approach after CNNs have been widely used is deep compression [20], which prunes specific weights whose magnitudes are below a threshold. Along this line of research, Liu et al. proposes to combine Winograd's minimal filtering algorithm in weight pruning [52]. Alternatively, Zhang et al. [96] use the alternating direction method of multipliers for weight pruning. As previously mentioned, weight pruning introduces unstructured sparsity to the network and thus cannot be implemented without specialized software and hardware [19].

On the other hand, channel pruning drops the entire filters in a CNN, which is more flexible. Traditional channel pruning approaches usually rank filters' importance and prune unimportant filters. For example, the magnitudes of filter weights are used as the criterion in [44] to rank the filter importance and those with small magnitudes are pruned out. First-order Taylor expansion is introduced in [64] to estimate the loss change when pruning each filter. [98] implements a spectral clustering algorithm to identify the efficient groups of filters that contribute essentially to the network performance. Geometric median [26] is used as a measurement to find the filters that contain the most redundancy. Thinet [57] proposes to identify the filters that result in minimal reconstructed error. [27] also introduces a similar idea with LASSO regression. GAL [51] leverages the idea of generative adversarial learning that learns to prune a network that mimics the output of the original baseline. HRank [49] proposes to prune filters with low-rank feature maps that contain less information.

Recently, several studies leverage RL in the network pruning area. N2N learning [3] proposes to use a recurrent neural network trained with RL for network compression. [89] and [97] use RL to prune connections in ResNet and DenseNet, respectively. AMC [25] introduces a deep deterministic policy gradient (DDPG) [47] agent to optimize the best number of filters in each convolutional layer and achieve considerable acceleration on mobile devices. A "try-and-learn" scheme is described in [32], which trains an agent that takes filter weights as the input and outputs a binary decision on whether the filters should be pruned. There are also a number of automatic pruning approaches related to our work [56, 45, 6, 40]. AutoPruner [56] uses the gradient information during fine-tuning to rank the filters so that pruning and fine-tuning can be combined. [6] uses Viterbi inference and [46] introduces to achieve such a purpose. Related to network pruning, there are also several works in the neural architecture search (NAS) area using RL to obtain compact CNNs [4, 99, 15]. Our proposed approach is related to AMC but is different in the following aspects. (1) AMC is *layer-wise* and extracts the features of each layer and the action is how many filters are be pruned in a layer. Our approach is *filter-wise*, which extracts the features of each filter and the action is which filter across all layers should be pruned. (2) AMC's search space is relatively small so that a multilayer perceptron can be used as the policy network without a search tree. However, the search space of filter-wise pruning is extremely large so we use MCTS to increase sample efficiency.

Monte Carlo tree search. MCTS was first proposed in 2006 which uses Monte Carlo rollouts to approximate the value of each state in a system that can be represented with a tree [36]. MCTS has been proved as an efficient algorithm in various sequence decision-making problems such as board [2] and real-time video games [67], complex materials design [58], and network virtualization [18].

However, for the tasks whose search spaces are extremely large, solely using MCTS does not work well due to the complexity of the task [76]. Here we formulate the channel pruning problem with RL by leveraging a neural network to predict the probabilities of pruning each filter in a CNN, taking a sequence of features extracted from the CNN as the input. MCTS is used for improving the policy with a lookahead search and helping the neural network make better decisions on filter selection. [14] proposes a related method with a straightforward binary lookahead search. Different from their work, we use MCTS combined with RL so that the search space is deeper and larger.

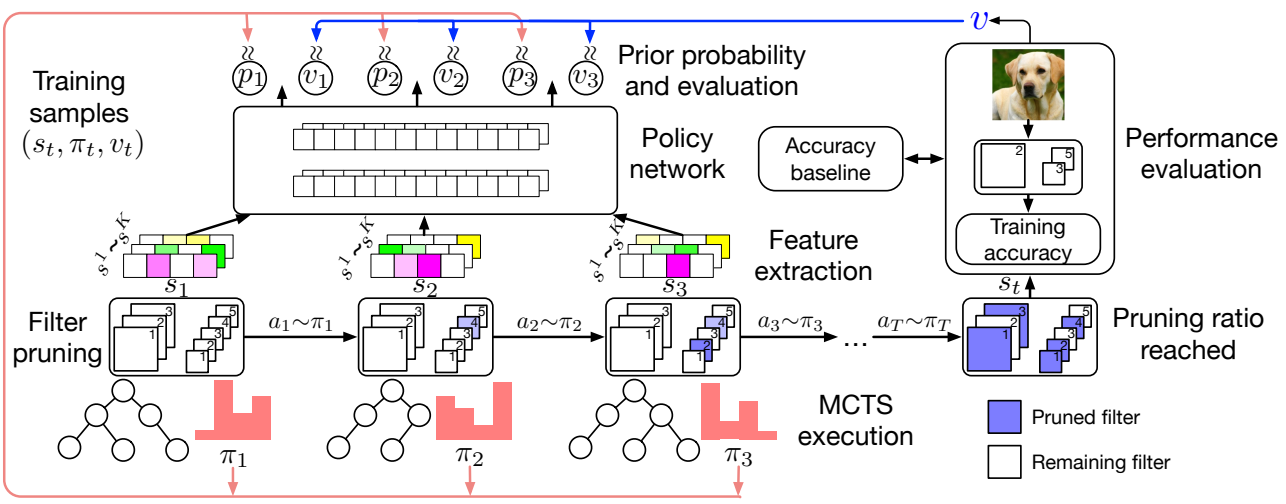


Figure 1. The overall framework of channel pruning using RL and MCTS. The neural network takes filter-related features as the input, and outputs (1) probabilities of filter selection and (2) a value as an estimate of the performance of the pruned network. MCTS simulations are executed to improve the sample efficiency for filter selection. When the pruning is done, the performance of the slimmed network is evaluated with training samples, which is used for the improvement of the performance estimation in the neural network.

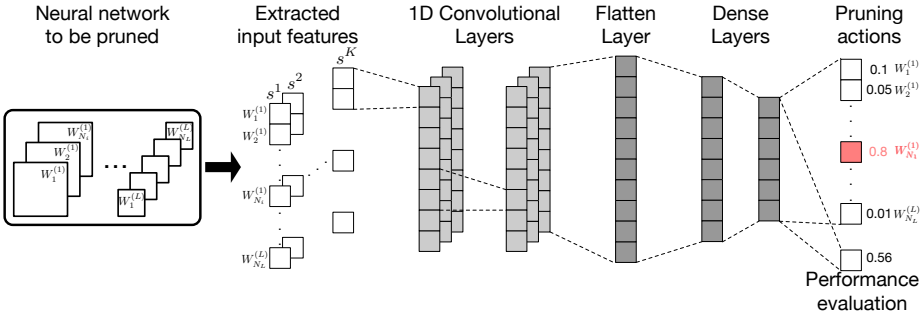


Figure 2. Architecture of the policy network. A sequence of features is extracted for each filter in the network to be pruned. 1D CNN has utilized to output the probabilities over filter selection and a value as the performance estimate of the slimmed network.

3. Network pruning using RL and MCTS

In this section, we present our approach in detail. We first formulate the channel pruning problem with RL and MCTS and present the detailed implementation of each part.

3.1. Notations and preliminaries

Suppose we have an L-layer CNN with N_i $(i = 1, 2, \cdots, L)$ filters in each layer. Denote $\mathbf{W}_j^{(i)}$ as the j-th filter in the i-th layer. The CNN's parameters $\mathcal{W}$ can be represented as $\mathcal{W} = \{\mathbf{W}^{(i)} \in \mathbb{R}^{N_{i+1} \times N_i \times h_i \times w_i},\ i = 1, 2, \cdots, L\}$, where N_i, N_{i+1}, h_i, w_i represent the numbers of input and output channels, filters' height and width in the i-th layer, respectively. Channel pruning aims to find an optimal set of parameters $\mathcal{W}'$ that leads to the least performance loss in the remaining network such that $|\mathcal{W}'| < B$, where $|\mathcal{W}'|$ corresponds to the number of elements and B is the maximum number of non-zero filters allowed in $\mathcal{W}'$. The performance loss can be measured with different criteria, such as training accuracy drop or absolute loss change.

3.2. Problem formulation and the proposed pipeline

We formulate the channel pruning problem as a sequence decision-making problem, which is modeled as a Markov decision process (MDP). The features extracted from the CNN to be pruned are represented as the state. When taking an action, i.e., pruning a filter, one network architecture transforms to another. A reward is given as an evaluation of the slimmed CNN. The goal of the system aims to find the slimmed CNN that maximizes the performance and meets a certain criterion (e.g., pruning a certain number of FLOPs).

The framework of our approach is described in Fig. 1 Starting with a CNN to be pruned, at each time step t, we extract a number of filter-related features s_t as the input of a neural network trained with RL for filter selection. The neural network outputs two components, i.e., (1) a value v_t as a performance estimate of the current slimmed CNN, and (2) a policy $\boldsymbol{p}_t$ that represents the probabilities of pruning each filter. $\boldsymbol{p}_t$ is improved by executing a lookahead search, i.e., MCTS, and the improved policy $\boldsymbol{\pi}_t$ is used to select the next action a_t, i.e., the next filter to be pruned. When reaching

the requirement, the pruned network's performance is evaluated with training samples and v is the final outcome of the evaluation. The tuples $(s_t, \boldsymbol{\pi}_t, v)$ are used for training the neural network via RL to improve its capability on filter selection and performance estimate.

3.3. Reinforcement learning for filter pruning

Feature extraction as state representation. Fig. 2 demonstrates the design of the neural network for pruning action selection trained with RL. Because the sizes of the filters $N_i \times h_i \times w_i$ are usually different from each other, we propose to extract features for each filter across all layers to get consistent dimensionality of feature representation. In our study, we use K features $s = \{s^i \in \mathbb{R}^{1\times N},\ i = 1, 2, \cdots, K\}$ to characterize each filter and combine them as a $K \times N$ matrix, where N is the total number of filters in the CNN to be pruned, as the representation of the state. In our study, the features are categorized into 3 types.

(1) Saliency responses. Saliency response has been proved as an effective criterion for the estimate of filter importance [68, 64, 29]. We feed training samples to the CNN to be pruned and record the value of each filter's feature map $\boldsymbol{Z}_j^{(i)}$ and gradient $\boldsymbol{G}_j^{(i)}$. Denote $z_{jk}^{(i)}$ and $g_{jk}^{(i)}$ the elements in $\mathbf{Z}_j^{(i)}$ and $\mathbf{G}_j^{(i)}$, respectively. We use 3 widely used quantities that are previously proposed to rank and prune filters as the components of our feature representation.

- Mean ReLU activation value [68], i.e., $s^1 = \{\frac{1}{|\boldsymbol{Z}_j^{(i)}|}\sum_k |z_{jk}^{(i)}|\}$, where $|\boldsymbol{Z}_j^{(i)}|$ is the the dimensionality of $\boldsymbol{Z}_j^{(i)}$ after vectorization and $i = 1, 2, \cdots, L,\ j = 1, 2, \cdots, N_L$.
- Average percentage of zero (APoZ) activation neurons [29], i.e., $s^2 = \{||\boldsymbol{Z}_j{}^{(i)}||_0\}$.
- Loss change estimate with first-order Taylor polynomial after each filter's removal [64], i.e., $s^3 = \{\frac{1}{|\boldsymbol{Z}_j^{(i)}|}\sum_k |z_{jk}^{(i)} \cdot g_{jk}^{(i)}|\}$.

(2) Filter weights. Because pruning filters with the smallest magnitude of weights has been widely used [44], we calculate the ℓ_2-norm of each filter's weights as a component of the feature sets, i.e., $s^4 = \{||\boldsymbol{W}_j^{(i)}||_2 = \frac{1}{|\boldsymbol{W}_j^{(i)}|}\sum_k (w_{jk}^{(i)})^2\}$, where $w_{jk}^{(i)}$ is the elements in $\boldsymbol{W}_j^{(i)}$, $|\boldsymbol{W}_j^{(i)}|$ is the dimensionality of $\boldsymbol{W}_j^{(i)}$ after vectorization.

(3) Structural-related quantities. Recent studies indicate that optimizing a network's structure also plays an essential role to achieve efficient compact neural networks [54, 60]. Therefore, it is also necessary to include the following structural-related features for characterizing each filter.

- Number of filters in the layer to which a filter belongs, in the original CNN to be pruned. $s^5 = \{\#\mathbf{W}_j^{(i)}\}$, where $\#\mathbf{W}_j^{(i)}$ denotes the number of filters in $\mathbf{W}^{(i)}$, i.e., $\#\mathbf{W}_j^{(i)} = N_i$.
- Number of filters in the layer to which a filter belongs, in the current slimmed CNN. $s^6 = \{\#\mathbf{W'}_j^{(i)}\}$.
- Index of the layer to which a filter belongs. $s^7 = \{\text{ind}(\mathbf{W}_j^{(i)})\}$, where $\text{ind}(\mathbf{W}_j^{(i)})$ is the layer index of $\mathbf{W}_j^{(i)}$, i.e., $\mathbf{W}_j^{(i)} = i$.

Filter selection. Since the input state of the neural network is a 2D tensor, 1D convolutional neural network becomes a natural choice. We use a 1D CNN f_θ parameterized by θ as the neural network for filter selection and performance estimate. f_θ takes as input the extracted features s_t described above and outputs a probability $\boldsymbol{p}_t$ and a scalar value v_t, i.e., $f_\theta(s_t) = (\boldsymbol{p}_t, v_t)$. $\boldsymbol{p}_t$ represents the probabilities of pruning each filter and v_t is an estimation of the performance of the current slimmed network. Because the CNN to be pruned usually contains thousands of filters, which makes the search space prohibitively large, we execute an MCTS search to get an improved policy $\boldsymbol{\pi}_t$ over $\boldsymbol{p}_t$ (which will be introduced in the next sub-section).

Model training. When the slimmed CNN reaches the predefined requirement, we use a simple strategy to evaluate its performance. Before pruning, we randomly prune filters from the CNN until the target is reached. We calculate the training accuracy (with the whole training set, or a subset if the number of training samples is too large) of the randomly slimmed CNN and use it as the initial baseline. When a filter is pruned, we compute and compare the training accuracy of the slimmed network with the baseline. If it outperforms the baseline, a reward of $+1$ is given and the baseline is updated with the new accuracy. Otherwise, a penalty of -1 is given. Therefore, the outcome $v \in \{-1, +1\}$, depending on whether a better-slimmed network architecture is found. At the end of each iteration, f_θ is trained with the tuples $(s_t, \boldsymbol{\pi}_t, v)$. With more training iterations, the baseline is gradually increased and more efficient slimmed CNN can be obtained. We use a slightly modified advantage actor-critic (A2C) approach [75, 61] for the RL training and the loss function of f_θ is defined in Eq. (1).

$$\mathcal{L}_\theta = (v_t - v)^2 - \boldsymbol{\pi}_t^T \cdot \log(\boldsymbol{p}_t) + c||\theta||^2, \qquad (1)$$

where c is a weight decay coefficient to prevent f_θ from overfitting.

3.4. MCTS for policy improvement

MCTS establishment. We introduce MCTS as a lookahead search mechanism to improve the filter pruning policy that is produced by f_θ. The workflow of MCTS execution is presented in Fig. 3. In an MCTS tree, each node represents a CNN configuration. Here we use the number of filters

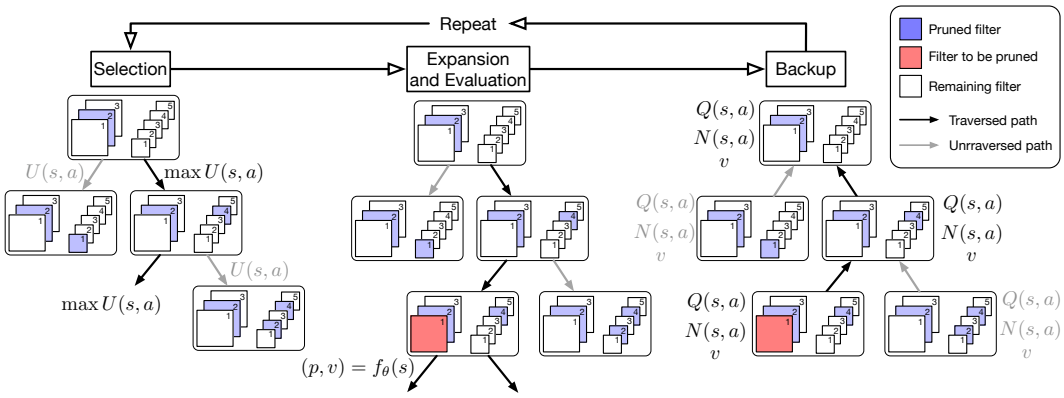


Figure 3. The workflow of MCTS execution for improving the pruning action selection. In each MCTS execution, the filter that maximizes the upper confidence bound $U(s,a)$ is selected. When reaching a leaf node, the neural network is used for filter selection instead of a rollout, and values of Q, N, and v are backpropagated.

in each convolutional layer as a unique representation for a certain CNN. A directed edge exists from Node n_i to n_j if n_j is obtained by removing a filter from n_i. For each node in the tree, we maintain the following three quantities.

- $Q(s,a)$. Expected reward for taking action a (pruning a filter) from state s (the current network architecture), i.e., the state-action value.
- $N(s,a)$. The number of times that an action a is selected from the state s, i.e., for a certain intermediate network architecture, how many times a filter has been chosen to remove during the MCTS.
- $P(s,\cdot)$. The prior probabilities of pruning each filter from f_θ by taking s as the input.

MCTS execution. Starting with a CNN to be pruned and an empty tree, the MCTS is iteratively executed by selecting actions that maximize the upper confidence bound $U(s,a)$, which is defined in Eq. (2) [77].

$$U(s,a) = Q(s,a) + c_{puct} \cdot P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}, \quad (2)$$

where c_{puct} is used for balancing exploration/exploitation.

When a leaf node is reached by continuously expanding nodes with Eq. (2), instead of performing a rollout, the leaf node is expanded with the output of the neural network f_θ, which generates both the probabilities $\boldsymbol{p}$ and a value v for performance estimation. Then for all traversed edges, v is backpropagated, the Q-value $Q(s,a)$ is updated by Eq. (3), and the visit count $N(s,a)$ is increased. If we encounter a terminal state s_t, we backpropagate the actual reward (+1 if the slimmed CNN outperforms the baseline, otherwise -1) rather than v.

$$Q(s,a) \leftarrow \frac{N(s,a) \cdot Q(s,a) + v}{N(s,a)+1}. \quad (3)$$

Therefore, the MCTS execution refers to a search process from the current node (the network to be pruned) to a leaf node with the help of the prior probabilities $\boldsymbol{p}$ from f_θ. We summarize the MCTS procedure in Algorithm 1.

Algorithm 1 The Monte Carlo Tree Search: MCTS(s_i)

Input: Raw configuration of the network to be pruned s_0, current configuration of the network to be pruned s_i, neural network for pruning action selection f_θ, pruning ratio γ, trainingAccBaseline b, $P(s,\cdot)$, c_{puct}.
Output: $N(s,a)$.
if FLOPs(s_i)/FLOPs(s_0) $< \gamma$ **then**
  if trainAcc(s_i) $> b$ **then**
    return 1
  else
    return -1
if s not visited **then**
  $P(s_i,\cdot), v = f_\theta(s_i)$
  return v
$maxU = -\infty$, bestAction $= -1$
for a in validActions(s_i) **do**
  $u = Q(s_i,a) + c_{puct} \cdot P(s_i,a) \cdot \sqrt{\frac{\sum_b N(s_i,b)}{1+N(s_i,a)}}$
  if $u > maxU$ **then**
    $maxU = u$, bestAction $= a$
s_{i+i} = pruneFilter(s_i, bestAction)
v = MCTS(s_{i+1})
$Q(s_i,a) = \frac{N(s_i,a) \cdot Q(s_i,a) + v}{N(s_i,a)+1}$
$N(s_i,a) += 1$
return v

After the MCTS search, the improved policy $\boldsymbol{\pi}$ can be simply obtained by normalizing the visit count $N(s,a)$ as shown in Eq. (4). $\boldsymbol{\pi}$ usually presents a much stronger policy compared to $\boldsymbol{p}$ and the policy network f_θ is trained to output probabilities that approximate $\boldsymbol{\pi}$.

$$\pi(a|s) = \frac{N(s,a)^{1/\tau}}{\sum_b N(s,b)^{1/\tau}}, \quad (4)$$

Approach	Test Acc.	Δ Acc.	Remaining FLOPs	Δ FLOPs
NISP [93]	NA → 99.18%	-	6.5E5	-71.6%
Min Weight [44]	99.22% → 98.88%	-0.34%	4.3E5	-81.2%
Taylor [64]	99.22 → 99.05%	-0.17%	3.9E5	-83.0%
SSL [88]	NA → 99.00%	-	2.0E5	-91.3%
GAL [51]	99.20 → 98.99%	-0.21%	1.0E5	-95.6%
RL-MCTS	99.22% → 99.28%	+0.06%	1.0E5	-95.6%

Table 1. Results of MNIST. Results of min weight and Taylor expansion are based on our own implementation.

where τ is the temperature coefficient that balances exploration and exploitation. Once the slimmed architecture meets the pre-defined requirements, we consider a whole self-play procedure is finished. Usually, we can empty the search tree after each self-play and perform the procedure for multiple times to obtain robust policies.

Reduction of search space. MCTS can significantly narrow down the search space compared to oracle pruning and improve the performance of the slimmed network. However, since a CNN usually contains thousands of filters, the search space is still large even with MCTS. In addition, many filters are commonly considered as important by various criteria so there is no need for search among them. Here we introduce two strategies to further reduce the search space of filter selection to boost the pruning process.

The first strategy is *mask*. We can mask out some important filters, which should not be pruned, before the MCTS execution to prevent MCTS from selecting the masked out filters. We can do this because [64] has shown that ranking filters in a CNN with first-order Taylor expansion can achieve a 0.8 Spearman's correlation [95] with the oracle ranking (1 means the two ranks are with full positive correlation and -1 implies full negative correlation) on various architectures. By masking out a certain percent of filters during the MCTS execution, the search space can be largely narrowed while only losing limited performance.

The second strategy is *segment*. We can split a task with a large pruning rate into several segments. By pruning a small number of FLOPs each time, fewer filters are pruned at one time, thus the MCTS search depth can be significantly reduced. When the network is not fine-tuned after each pruning segment, the approach is within the scope of single-shot pruning. If the network is fine-tuned, it becomes a progressive pruning approach.

4. Experiments

4.1. Setup

We evaluate our approach with the following experiments: LeNet5 [39] on MNIST [39], VGG-16 [78] and ResNet-56 [22] on CIFAR-10 [37], and ResNet-50 on ILSVRC-2012 (ImageNet) [13]. A VGG16 style 1D CNN with a kernel size of 3 in each convolutional layer is used for the action selection and performance estimate. We perform 50 MCTS from each state to obtain the improved policy that prunes a filter from the current network, and 5 self-plays for each pruning iteration. The weight decay factor c is set to $1e-4$. The scaling factors c_{puct} and τ that balance the exploration and exploitation are set to 1 for the sake of simplicity. The MCTS is executed with a root parallelization scheme. A queue with a maximum length of 5000 is used to collect the training tuples $(s, \boldsymbol{\pi}, v)$. The earliest samples are replaced if the training samples exceed the maximum queue length. After each pruning iteration, the policy network f_θ is trained for 100 epochs with an Adam optimizer (learning rate $4e^{-5}$, batch size 64).

We select 5000 samples from each training set for performance evaluation to prevent overfitting. Before pruning, we randomly prune filters from the CNN until the pre-defined target is reached. The performance of this slimmed network is used as the initial baseline. In this study, we use the *training accuracy drop* as the evaluation criterion. We also discuss other criteria in the ablation study section. After the training process, we use the selected samples to evaluate the performance of the slimmed network. If the slimmed network outperforms the previous model, the baseline (accuracy) is updated with the new number, and the whole process repeats. If the performance cannot be further improved in 3 continuous iterations, we stop the current training process, prune the selected filters, and fine-tune the slimmed network for 120 epochs, with a learning rate of 0.1, which is divided by 10 every 30 epochs. If the *segment* strategy is used, we fine-tune the slimmed network for 30 epochs and re-calculate the features of each filter with the whole training set before entering the next pruning segments. The pruned filters are also masked out so that they will not be selected in the following iterations. When the pruning is done, we fine-tune the remaining network for another 30 epochs with a batch size of 256, starting with a learning rate of $1e^{-4}$, which is divided by 5 at epoch 15.

4.2. Result comparison

LeNet5 on MNIST. We first evaluate our approach with LeNet5 on the MNIST dataset. In our LeNet5 implementation, the network contains two convolutional layers, with 20 and 50 filters in each layer. We first train the model from scratch and achieve a test accuracy of 99.22%. As shown in Table 1, our approach prunes 95.6% of the total FLOPs,

Model	Approach	Test Acc.	Δ Acc.	Remaining FLOPs	Δ FLOPs
VGG-16	Min Weight [44]	93.25% → 93.40%	+0.15%	2.06E8	-34.2%
	FPGM [26]	93.58% → 93.54%	-0.04%	2.06E8	-34.2%
	SSS [33]	93.96% → 93.02%	-0.94%	2.00E8	-39.6%
	GAL [51]	93.96% → 93.42%	-0.54%	1.72E8	-45.2%
	ABCPruner* [50]	93.02% → 93.08%	+0.06%	0.82E8	-73.7%
	AOFP* [14]	93.38% → 93.28%	-0.10%	0.77E8	-75.3%
	RL-MCTS	93.51% → 93.90%	+0.39%	1.71E8	-45.5%
		93.51% → 93.72%	+0.21%	0.67E8	-79.9%
ResNet-56	Min Weight [44]	93.53% → 93.30%	-0.23%	7.8E7	-38.6%
	NISP [93]	93.53% → 93.38%	-0.15%	7.1E7	-43.8%
	GAL [51]	93.50% → 92.74%	-0.76%	6.5E7	-48.5%
	AMC* [25]	-	-0.90%	6.3E7	-50.0%
	TAS* [15]	94.46% → 93.69%	-0.77%	6.0E7	-52.7%
	LFPC [23]	93.59% → 93.24%	-0.35%	5.9E7	-52.9%
	FPGM [26]	93.68% → 93.74%	+0.06%	5.9E7	-53.2%
	ABCPruner* [50]	93.26% → 93.23%	-0.03%	5.8E7	-54.1%
	RL-MCTS	93.20% → 93.56%	+0.36%	5.7E7	-55.0%

Table 2. Results of VGG and ResNet on CIFAR-10. * indicates automated pruning approach.

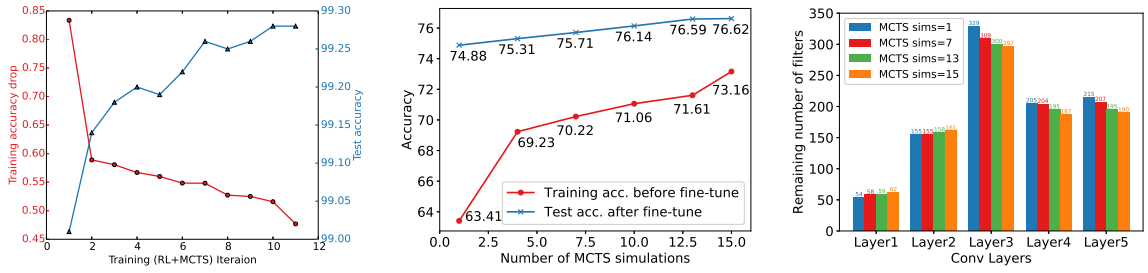
Approach	Top-1 Acc.	Top-5 Acc.	Δ Top-1 Acc.	Δ Top-5 Acc.	Δ FLOPs
SFP [24]	76.15% → 74.61%	92.87% → 92.06%	-1.54%	-0.81%	-41.8%
FPGM [26]	76.15% → 75.50%	92.84% → 92.63%	-0.65%	-0.21%	-42.2%
GAL [51]	76.15% → 71.95%	92.87% → 90.94%	-4.20%	-1.93%	-43.0%
TAS* [15]	77.46% → 76.20%	93.55% → 93.07%	-1.26%	-0.48%	-43.5%
Slim NN [92]	76.00% → 74.90%	-	-1.10%	-	-43.8%
HRank [49]	76.15% → 71.95%	92.87% → 92.33%	-1.17%	-0.54%	-43.8%
Taylor [64]	76.18% → 74.98%	-	-1.68%	-	-44.9%
MetaPrunning* [53]	76.01% → 72.17%	-	-3.84%	-	-45.3%
AutoPruner* [56]	76.15% → 74.76%	92.87% → 92.15%	-1.39%	-0.72%	-51.2%
LFPC [23]	76.15% → 74.46%	92.87% → 92.04%	-1.69%	-0.83%	-53.5%
ABCPruner* [50]	76.01% → 73.86%	92.96% → 91.69%	-2.15%	-1.27%	-54.3%
Random Search	77.34% → 73.29%	93.27% → 90.98%	-4.05%	-2.29%	-45.0%
RL-MCTS	77.34% → 76.80%	93.27% → 93.00%	-0.54%	-0.27%	-46.1%
	77.34% → 76.46%	93.37% → 92.83%	-0.88%	-0.34%	-55.0%

Table 3. Results of ResNet-50 on ILSVRC-2012. * indicates automated pruning approach.

which results in an extremely compact network with only 2 and 15 filters remaining in each layer, and the test accuracy can still increase by 0.06% after fine-tuning. The previous best performance (NISP [93]) is 99.18%, by pruning 71.6% FLOPs. By pruning comparable FLOPs, GAL [51] achieves an accuracy of 98.99%, while ours is 99.28%. These results validate the improvement with our proposed approach over the state of the art on MNIST.

VGG and ResNet on CIFAR-10. Table 2 presents the results of pruning VGG16 and ResNet-56 on CIFAR-10. For both architectures, we mask 70% of the unimportant filters with the *mask* strategy and prune 1% of the total FLOPs with the *segment* strategy. Our approach can reduce nearly 80% of the total FLOPs on VGG16 with no performance loss, which outperforms previous studies with a clear margin. In particular, AOFP [14], which uses a similar binary search idea, achieves a test accuracy of 93.28% by pruning 75.3% FLOPs, with 0.1% performance loss. On ResNet-56, our pruned model achieves a test accuracy of 93.56%, which is 0.36% higher than the baseline, by removing 55% FLOPs, which performs better than recent automated pruning approaches (TAS [15] and ABCPruner [50]).

ResNet-50 on ILSVRC-2012. We finally conduct experiments with ResNet50 on the ILSVRC-2012 dataset to validate the effectiveness of our approach on large-scale network architecture and datasets. To alleviate computation cost, we first rank all filters with the Taylor expansion criterion [64] and get an importance score for each filter. We first prune 30% of the total FLOPs based on the filter importance and mask the least 80% important filters. We split the whole pruning process into a sequence of segments in which 0.5% FLOPs are pruned. The results are presented in Table 3. By pruning 46% FLOPs, the networks' top-1 and top-5 accuracies only drop by 0.54% and 0.27%. When



(a) LeNet5 on MNIST (b) AlexNet on CIFAR-10 (c) AlexNet on CIFAR-10

Figure 4. Effectiveness with different numbers of training iterations and MCTS executions. (a) With more training iterations, the pruned network's training accuracy before fine-tuning and test accuracy after fine-tuning increases. (b) With more MCTS simulations, the performance is improved, and (c) more filters in deeper layers are pruned.

Model	Acc. drop	Acc. change	Loss drop	Loss change
AlexNet on CIFAR-10	76.62%	76.54%	76.55%	76.40%
VGG-16 on CIFAR-10	93.90%	93.68%	93.82%	93.85%

Table 4. Performance comparison using different evaluation criteria after pruning.

the pruning ratio increases to 55%, the top-1 accuracy of the slimmed network is 76.46%, which is only 0.88% lower than the baseline. These results show competitive performance compared to existing channel pruning approaches. To understand the effectiveness of our approach, we also prune the network with random search, using the same number of executions as our approach. It turns out that the performance of random search is unsatisfactory, which drops more than 4% top-1 accuracy when pruning 45% flops.

4.3. Ablation study

Performance over iteration. We first show that the performance is improved with more training iterations. We use the LeNet5 on the MNIST case for illustration. From the experiment results (Fig. 4(a)) we observe that the drop of the training accuracy decreases as more iterations are conducted. After fine-tuning the slimmed models, the test accuracy increases to 99.28% after 10 iterations.

Effect on the number of MCTS simulations. To study the effect of the number of MCTS simulations on the performance, we conduct experiments with different numbers of MCTS simulations for action selection. To compare the number of filters in each layer of the slimmed network, we use AlexNet, which contains five convolutional layers, on CIFAR-10 as an example. We plot the training accuracy before fine-tuning and the test accuracy after fine-tuning in Fig. 4(b). Not surprisingly, as more MCTS are executed each time, better performance can be obtained. The test accuracy increases significantly as the number of MCTS simulations grows from 1 to 10. When more than 10 MCTS simulations are executed, the performance improvement gets flattened, which indicates that the slimmed network gets close to the optimum.

We further plot the network architectures searched with different numbers of MCTS simulations in Fig. 4(c). As more MCTS are executed, more filters in the deeper layers are pruned. This finding indicates that these filters play less essential roles compared to those in the shallower layers, which is consistent with the observations in several other studies [44, 64, 85].

Effect on the model evaluation criteria. We also investigate whether different criteria for evaluating the slimmed model after each training iteration have any influence on the performance. We use the following four criteria for comparison: (1) accuracy drop, (2) accuracy change, i.e., the absolute value of (1), (3) loss drop, and (4) loss change, i.e., the absolute value of (3). From the results shown in Table 4 we observe that using training accuracy as the criteria performs slightly better than using loss. However, the performance of the slimmed networks with different criteria is generally similar, which implies that our approach is not sensitive to these commonly used evaluation criteria.

5. Conclusion

We proposed a pruning approach based on reinforcement learning, incorporated with a lookahead search mechanism via MCTS. A neural network that takes a sequence of filter-related features as the input is trained with RL to output the probabilities of pruning each filter in a CNN. An MCTS is executed for each pruning action to increase the sample efficiency in the RL training. We implemented our approach with various benchmark networks and experiments showed the effectiveness of our approach. We will focus on improving the mask and the evaluation strategies to further increase the performance and investigate the potential usage of this approach in the zero-shot pruning manner in the future.

References

[1] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[2] Broderick Arneson, Ryan B Hayward, and Philip Henderson. Monte carlo tree search in hex. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):251–258, 2010.

[3] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. In *International Conference on Learning Representations*, 2018.

[4] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[5] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[6] Maxim Berman, Leonid Pishchulin, Ning Xu, Matthew B Blaschko, and Gérard Medioni. Aows: Adaptive and optimal network width search with latency constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11217–11226, 2020.

[7] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[8] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pages 742–751, 2017.

[9] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[10] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[11] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[12] Yehuda Dar, Vidya Muthukumar, and Richard G Baraniuk. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning. *arXiv preprint arXiv:2109.02355*, 2021.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive cnn width optimization. In *International Conference on Machine Learning*, pages 1607–1616, 2019.

[15] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *Advances in Neural Information Processing Systems*, pages 759–770, 2019.

[16] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[18] Soroush Haeri and Ljiljana Trajković. Virtual network embedding via monte carlo tree search. *IEEE transactions on cybernetics*, 48(2):510–521, 2017.

[19] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 243–254. IEEE, 2016.

[20] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[21] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2009–2018, 2020.

[24] 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*, pages 2234–2240, 2018.

[25] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[26] 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*, pages 4340–4349, 2019.

[27] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[29] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[30] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[31] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[32] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 709–718. IEEE, 2018.

[33] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

[34] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[35] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

[36] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[37] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] Eugene Lee and Chen-Yi Lee. Neuralscale: Efficient scaling of neurons for resource-constrained deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1478–1487, 2020.

[41] Howard Lee and Yi-Ping Phoebe Chen. Image based computer aided diagnosis system for cancer detection. *Expert Systems with Applications*, 42(12):5356–5365, 2015.

[42] Chengcheng Li, Zi Wang, and Hairong Qi. Fast-converging conditional generative adversarial networks for image synthesis. In *2018 25th IEEE international conference on image processing (ICIP)*, pages 2132–2136. IEEE, 2018.

[43] Chengcheng Li, Zi Wang, and Hairong Qi. An efficient pipeline for pruning convolutional neural networks. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 907–912. IEEE, 2020.

[44] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[45] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8018–8027, 2020.

[46] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 608–624. Springer, 2020.

[47] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[48] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[49] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2020.

[50] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.

[51] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019.

[52] Xingyu Liu, Jeff Pool, Song Han, and William J Dally. Efficient sparse-winograd convolutional neural networks. In *International Conference on Learning Representations*, 2018.

[53] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3296–3305, 2019.

[54] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2018.

[55] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[56] Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, page 107461, 2020.

[57] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[58] Thaer M. Dieb, Shenghong Ju, Kazuki Yoshizoe, Zhufeng Hou, Junichiro Shiomi, and Koji Tsuda. Mdts: automatic complex materials design using monte carlo tree search. *Science and technology of advanced materials*, 18(1):498–503, 2017.

[59] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[60] Deepak Mittal, Shweta Bhardwaj, Mitesh M Khapra, and Balaraman Ravindran. Recovering from random pruning: On the plasticity of deep convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 848–857. IEEE, 2018.

[61] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[62] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[63] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[64] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[65] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

[66] Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

[67] Tom Pepels, Mark HM Winands, and Marc Lanctot. Real-time monte carlo tree search in ms pac-man. *IEEE Transactions on Computational Intelligence and AI in games*, 6(3):245–257, 2014.

[68] Adam Polyak and Lior Wolf. Channel-level acceleration of deep face representations. *IEEE Access*, 3:2163–2175, 2015.

[69] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[70] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.

[71] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[72] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[73] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[74] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[75] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[76] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[77] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[78] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[79] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[80] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[81] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[82] Zi Wang. Data-free knowledge distillation with soft targeted transfer set synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10245–10253, 2021.

[83] Zi Wang. Learning fast converging, effective conditional generative adversarial networks with a mirrored auxiliary classifier. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2566–2575, 2021.

[84] Zi Wang. Zero-shot knowledge distillation from a decision-based black-box model. *arXiv preprint arXiv:2106.03310*, 2021.

[85] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14913–14922, 2021.

[86] Zi Wang, Chengcheng Li, Xiangyang Wang, and Dali Wang. Towards efficient convolutional neural networks through low-error filter saliency estimation. In *Pacific Rim International Conference on Artificial Intelligence*, pages 255–267. Springer, 2019.

[87] Zi Wang, Dali Wang, Chengcheng Li, Yichi Xu, Husheng Li, and Zhirong Bao. Deep reinforcement learning of cell movement in the early stage of c. elegans embryogenesis. *Bioinformatics*, 34(18):3169–3177, 2018.

[88] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[89] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[90] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[91] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 2130–2141, 2019.

[92] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[93] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[94] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017.

[95] Jerrold H Zar. Significance testing of the spearman rank correlation coefficient. *Journal of the American Statistical Association*, 67(339):578–580, 1972.

[96] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

[97] Xuanyang Zhang, Zhanxing Zhu, Zenglin Xu, et al. Learning to search efficient densenet with layer-wise pruning. 2018.

[98] Huiyuan Zhuo, Xuelin Qian, Yanwei Fu, Heng Yang, and Xiangyang Xue. Scsp: Spectral clustering filter pruning with soft self-adaption manners. *arXiv preprint arXiv:1806.05320*, 2018.

[99] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.