# VOTE FOR NEAREST NEIGHBORS META-PRUNING OF SELF-SUPERVISED NETWORKS


## ABSTRACT

Pruning plays an essential role in deploying deep neural nets (DNNs) to the hardware of limited memory or computation. However, current high-quality iterative pruning can create a terrible carbon footprint when compressing a large DNN for a wide variety of devices and tasks. Can we reuse the pruning results on previous tasks to accelerate the pruning for a new task? Can we find a better initialization for a new task, e.g., a much smaller network closer to the final pruned model, by exploiting its similar tasks? We study this "nearest neighbors meta-pruning" problem by first investigating different choices of pre-trained models for pruning under limited iterations. Our empirical study reveals several advantages of the self-supervision pre-trained model when pruned for multiple tasks. We further study the overlap of pruned models for similar tasks and how the overlap changes for different layers. Inspired by these discoveries, we develop a simple but strong baseline "Meta-Vote Pruning (MVP)" that significantly reduces the pruning iterations for a new task by initializing a sub-network from the pruned models for tasks similar to it. In experiments, we demonstrate the advantages of MVP by extensive empirical studies and comparisons with popular pruning methods.


## 1 INTRODUCTION

Deep learning often requires to train an over-parameterized model with millions of parameters for promising generalization performance. The computation and memory of modern GPUs or clusters can support to train large-scale models but directly deploying them to edge devices can easily violate the hardware limits on memory and computation. Network pruning (Han et al., 2016; Tian et al., 2020; Li et al., 2020; Chin et al., 2020) has been widely studied to compress neural nets by removing redundant connections (weights) and/or nodes (neurons). Numerous empirical results have verified that pruning can compress the original network to much smaller sub-networks that still enjoy comparable (or even better) performance. Instead of reducing the network to the target size by one-time pruning, iterative pruning that alternates between pruning and fine-tuning for iterations usually achieves better performance (Han et al., 2015; Li et al., 2017). Theoretically, a line of recent works (Frankle & Carbin, 2018; Ye et al., 2020; Savarese et al., 2020; Malach et al., 2020) attempts to prove the lottery ticket hypothesis, i.e., the existence of such sub-networks, for different pruning settings.

In a variety of practical applications, a pre-trained network usually needs to be pruned and customized for a wide variety of devices and tasks. Running an iterative pruning algorithm for every device or task from a large pre-trained network can create enormous carbon footprint overload in our biosphere and waste a lot of computational power. Moreover, it is not uncommon that the data available on each device or task are insufficient for pruning a large-scale network to a reliable sub-network. Can we reuse the pruning results achieved on previous tasks as prior knowledge to reduce the computation and data required by pruning on new tasks? We call this problem "non-parametric meta-pruning". In this paper, we mainly focus on a special case of meta-pruning that initializes a sub-network for a given new task, which is extracted from a pre-trained model using the pruned models for previous tasks. Meta-pruning is non-parametric if no parametric model is trained to produce the initialization. It is analogous to MAML (Finn et al., 2017) in that the meta-objective optimizes the initialization but does not directly control the model update afterward. It differs from MAML in that both the sub-network's architecture and weights need to be initialized and the initialization is not universal but task-specific.

Since meta-pruning aims to find better initialization for pruning, to strengthen the influence of initialization on the final pruned model, we keep both the number of iterations and the learning rate small. Restricting the number of iterations also controls the computational cost and carbon footprint of meta-pruning much less than conventional pruning that may require many iterations. Under these two constraints, the choice of pre-trained model is critical to the meta-pruning performance because

(1) it needs to provide initialization sub-networks for different (even unseen) tasks; and (2) a few iterations with slight fine-tuning to the sub-networks should suffice to produce high-quality pruned models for targeted tasks. This raises new challenges since various pruning methods start from a model pre-trained for the target task, which is not available in meta-pruning. In fact, meta-pruning follows a more practical setting that one single pre-trained model is tailored for different tasks using limited iterations. We compare two kinds of widely used pre-trained models, i.e., one is trained by supervised learning on a labeled dataset, and the other is trained by self-supervised learning (Grill et al., 2020; Chen et al., 2020a; Zbontar et al., 2021) on unlabeled data.

The primary contribution of this paper is two folds. In the first part, we conduct a thorough empirical study of iterative pruning applied to hundreds of different tasks when using the two pre-trained models. No meta-pruning is used in this part and its main purpose is (1) to compare the two types of pre-trained models for different tasks' pruning and (2) to find the connection among pruned models for different but similar tasks. We thus build a dataset of tasks and their corresponding models pruned from the two pre-trained models. Statistics and evaluations on this dataset indicate several advantages of self-supervision pre-trained model for meta-pruning. Moreover, more similar tasks tend to share more nodes/filters preserved in their pruned models, especially in deeper layers that notably capture high-level task-specific features. We further discover a strong correlation between the chance that a node/filter is retained after the pruning for a task and the number of times it is also preserved in pruned models for similar tasks, which sheds insights to the development of meta-pruning algorithms.

Motivated by above empirical study, the second part of this paper proposes a simple yet strong meta-pruning baseline called "meta-vote pruning (MVP)" that can significantly reduce the pruning cost, memory and data required by previous pruning approaches yet still produce pruned models with promising performance. Given a self-supervision pre-trained model, MVP finds a sub-network for a new task by selecting nodes/filters through majority voting among similar tasks, i.e., we sample nodes/filters according to their chances being selected into the pruned model for similar tasks. To keep the baseline simple, we sample the same proportion of nodes/filters as the targeted pruning ratio and then apply a few epochs of fine-tuning with a small learning rate using training data of the targeted task. Although more sophisticated procedure can be developed, the proposed baseline already saves a substantial amount of computation and memory while still maintains a high test accuracy of the pruned models. We demonstrate this via experiments over tasks drawn from CIFAR (Krizhevsky & Hinton, 2009) and ImageNet (Deng et al., 2009). Moreover, when reducing the training data available for targeted tasks, MVP suffers much less performance degeneration of the pruned models than iterative pruning without leveraging any meta knowledge.

## 2 Related works

**Network pruning** Network pruning has been widely studied to compress network and accelerate its inference for a single task. We mainly summarize node/filter pruning below. To encourage the sparsity of the pruned network, $L_0$ (Louizos et al., 2018), $L_1$ (Liu et al., 2017) or $L_2$ (Han et al., 2015) regularization have been used, and recent polarization regularization (Zhuang et al., 2020) shrinks some nodes towards 0 and meanwhile strengthen the others to keep important nodes intact. Different criteria have been proposed to evaluate the importance of nodes/filters. Li et al. (2017) prunes the filters with the smallest sum of parameters' absolute values. Lin et al. (2018) prune filters according to the second-order Taylor expansion of the loss. They excel on single-task pruning but require more computations and data than meta-pruning setting because: (1) a large model needs to be pre-trained for every task; (2) the pruning starts from the large pre-trained model and can be computational costly; and (3) more data is required to prune a large model.

**Self-supervised learning** Self-supervised learning can learn rich and universal representations from unlabeled data. Comparing to supervised learning, it does not rely on labeled data that are usually expensive to collect in practice. Moreover, it can be trained once and applied to different tasks and thus creates less carbon footprint. Recent self-supervised learning methods (He et al., 2020; Chen et al., 2020a) show great potential in approaching or even surpassing supervised learning performance on a variety of tasks, especially when used to train a large network (Chen et al., 2020b). Hence, how to prune and re-use self-supervised networks for many different tasks is a critical open problem. Knowledge distillation (Hinton et al., 2015) of data similarity has been studied in (Koohpayegani et al., 2020) to compress a self-supervised network to a smaller student model. Network pruning

methods for supervised networks has also been applied to self-supervised networks (Caron et al., 2020). However, they mainly focus on compressing the original network rather than pruning it for downstream tasks, which is a main topic of this paper. We will explore several advantages of self-supervised networks in meta-pruning.

**Meta-pruning** To the best of our knowledge, the non-parametric meta-pruning problem, i.e., how to prune a model for a target task using the pruned models of other tasks, has not been specifically studied in previous work. However, several recent researches aim at learning meta(prior) knowledge that can improve pruning in different scenarios. MetaPruning (Liu et al., 2019) trains a weight-generation meta-network to prune the same network for the same task under different constraints, e.g., user/hardware defined pruning ratios. DHP (Li et al., 2020) addresses the same problem but does not rely on any reinforcement learning or evolutionary algorithm since it makes the pruning procedure differentiable. Meta-learning has been studied to find better weight-initialization for pruning on different tasks, e.g., Tian et al. (2020) applies Reptile (Nichol et al., 2018) for overfitting reduction. Meta-learner has also been trained to select the most appropriate pruning criterion for different tasks (He et al., 2019). In (Sun et al., 2020), a sparse shared backbone network is jointly trained for multiple tasks but it cannot be adapted to new tasks. The main difference of our approach to them are: (1) we do not train an extra meta-learner but instead use majority voting to directly relate training tasks to target tasks; and (2) our meta-voting generates a pruned sub-network to initialize the target task training, which is simple but significantly reduces the training cost afterwards.

## 3 Empirical Study: Pre-trained model pruning for Different Tasks

**Choices of the Pre-trained Model** One motivation behind meta-pruning is avoiding to pre-train a large model for every task. Instead, we use one single pre-trained model for all tasks and meta-pruning aims to pick a sub-network from it for every task as initialization. There are two major choices for the pre-trained model: (1) a neural net trained by supervised learning on a labeled dataset covering many diverse classes, e.g., ImageNet; (2) a neural net trained by self-supervised learning on unlabeled data that are widely available or can be collected with much lower cost. In practice, we can easily find them available online and do not need to pay computation for the pre-training. In the empirical study, we will compare them when pruned for different tasks with limited budget on the pruning iterations. Specifically, we adopt ResNet-18 (He et al., 2016) and ResNet-50 as the network architectures to pre-train on two datasets, CIFAR-100 (Krizhevsky & Hinton, 2009) and Tiered-ImangeNet (Ren et al., 2018), respectively. For each dataset, we compare the two types of pre-trained models mentioned above, whose training follows (Devries & Taylor, 2017) and SimSiam (Chen & He, 2020) (only the encoder is used), respectively.

**Algorithm 1** Iterative Filter Pruning (IFP)

**Input** : Pre-trained network $F(\cdot;\theta)$, Task $T$ and its training set $D_T$, $K$, Pruning hyperparameters $J, h, r, p$

**Initialize** : $\Omega_\ell \leftarrow [n_\ell]$, the set of filters preserved in layer-$\ell$

Replace the last fully-connected layer $\theta_L$ of $F(\cdot;\theta)$ with a newly initialized one for the targeted task $T$;

Train $\theta_L$ for $K$ epochs on $D_T$ with other layers fixed;

**for** $j \leftarrow 1$ ***to*** $J$ **do**

  **for** $\ell \leftarrow 1$ ***to*** $L-1$ **do**

    **if** $j\%h = 0$ *and* $|\Omega_\ell| > (1-r)n_\ell$ **then**

      Prune $p\%$ of filters in $\Omega_\ell$ that have the smallest activation values over $D_T$, i.e., $\frac{1}{|D_T|}\sum_{x\in D_T} f_{\ell,i}(x)$;

    **end**

  **end**

  Apply one SGD step on a mini-batch of $D_T$ to fine-tune the remained filters $\{\theta_{\ell,i} : \ell \in [L-1], i \in \Omega_\ell\}$ and $\theta_L$;

**end**

Fine-tune the pruned model for one epoch on $D_T$;

**Pruning Algorithm** In contrast to magnitude/score-based pruning (Li et al., 2017) with one-time selection of nodes/weights, iterative pruning alternates between network pruning and fine-tuning of model weights for multiple iterations, each of which prunes $p\%$ of the remaining nodes/weights so it gradually prunes a large-network to the targeted size. It usually achieves better performance in empirical comparisons with other pruning methods and has also been mainly studied in theoretical works about Lottery Ticket Hypothesis (Frankle & Carbin, 2018). This empirical study is conducted on convolutional neural nets so we apply iterative filter-pruning (IFP) that removes filters with the

smallest activation values averaged over all training samples. Similar methods are used in (Tan & Motani, 2020). The detailed procedure is described in Algorithm 1. Given a pre-trained network $F(\cdot;\theta)$ of $L$ layers (layer-$L$ is fully-connected) with parameter $\theta = \{\theta_\ell\}_{\ell=1:L}$ and a training set $D_T$ of a target task $T$, let $\theta_\ell = \{\theta_{\ell,i}\}_{i=1:n_\ell}$ denote all parameters in layer-$\ell$ composed of $\theta_{\ell,i}$ for every filter-$i$, IFP starts to train a new linear classifier (i.e., $\theta_L$) attached to the penultimate-layer of $F(\cdot;\theta)$ for $K$ epochs on $D_T$ and follows by $J$ pruning iterations. It prunes $p\%$ of the filters remained in each layer every $h$ iteration according to their activation values $f_{\ell,i}(x)$ and fine-tune the pruned model in every iteration. It stops to prune layer-$\ell$ if reaching the targeted pruning ratio $r$.

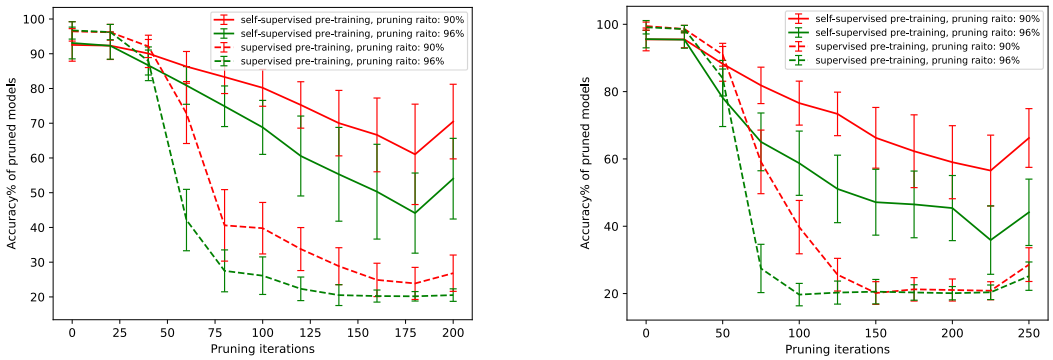



Figure 1: Supervised vs. self-supervised network for pruning (Alg. 1) with fewer iterations: test accuracy (mean±std) of pruned models with 200 and 250 pruning iterations for 310 tasks drawn from CIFAR-100 (**LEFT**) and Tiered-ImageNet (**RIGHT**).

**A Dataset of Pruned Models** Our empirical study is carried out on CIFAR-100 and Tiered-ImageNet. For each dataset, we randomly draw 310 classification tasks, each defined on 5 classes sampled without replacement. Each class from CIFAR-100 and Tiered-ImageNet has 500 and 1300 samples, respectively. Running Algorithm 1 for all 310 tasks using two different pre-trained models creates a dataset of pruned models. For each task $i$, we record its classes $C_i$, the set of preserved filters $\{\Omega_\ell\}_{\ell=1:L-1}$ and the pruned model $\theta_T$. We use the same hyperparameters for different tasks. As argued in Sec. 1, meta-pruning targets limited iterations and small learning rate in order to keep the computational cost low and the initialization matter more. Hence, when comparing the pre-trained models (Sec. 3.1), we use a learning rate of $0.0001$, $K = 10$, $J = 200(250)$, $h = 60(75)$, batch-size of $128(256)$ for CIFAR-100 (Tiered-ImageNet) and $p = 1 - \sqrt[\lfloor J/h \rfloor]{1-r}$ so the targeted pruning ratio $r$ is achieved after iteration-$(J-h)$. We have tried two pruning ratios $r = \{0.9, 0.96\}$. To study the similarity between the "winning tickets" for different tasks (Sec. 3.2-3.3), we use more delicately pruned models achieved by running more iterations ($J = 800(1000)$) in above experiments.

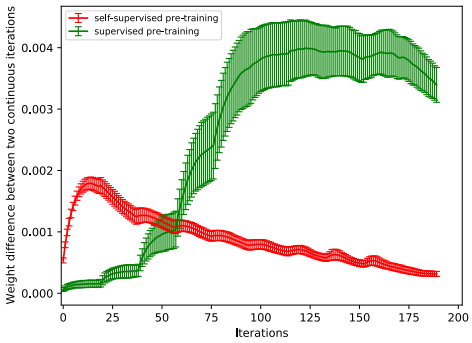


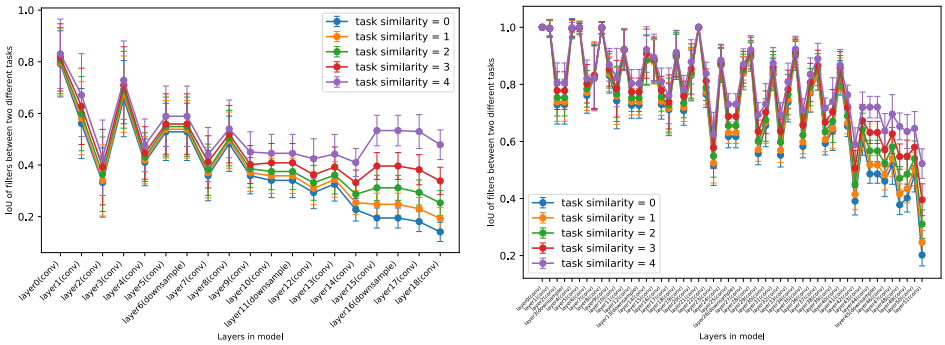

Figure 2: Absolute difference ($\ell_1$-distance) of model weights (mean±std) between two consecutive pruning iterations on CIFAR-100.

Since this dataset covers different tasks and their pruned/fine-tuned models from different pre-trained networks, it may be of independent interest to other problems beyond network pruning, e.g., transfer/multi-task/meta learning, fine-tuning, self-supervision, interpretable machine learning, etc.

### 3.1 Which Pre-trained Model to Prune? Supervised vs. Self-Supervised Model

We compare supervised and self-supervised networks when pruned for 310 different tasks drawn from CIFAR-100 (Tiered-ImageNet) using IFP (Alg. 1) with fewer iterations ($J = 200(250)$) with small learning rate ($0.0001$). We report how the test accuracy (mean±std) changes over pruning iterations in Fig. 1. The accuracy decreases for all the cases as more filters removed from the network. It increases in the last epoch due to the fine-tuning in line 11 of Alg. 1. For both datasets and pruning ratios, self-supervised network consistently outperforms supervised network when being used as the

pre-trained model for pruning. We posit the reason is that the the filters/nodes in self-supervised networks are more disentangled across different classes, i.e., a filter corresponds to fewer classes than that in supervised networks. Hence, it requires fewer iterations and smaller learning rate to fine-tune the filters for the targeted task's classes. e.g., it might only need to adjust the overall scales of pre-trained filters. So pruning a self-supervised network for a new task can be easier and starts closer to the final model. To further verify this conjecture, in Fig. 2, we study how the filter weights change between two consecutive pruning iterations applied to the two pre-trained models. Initialized with a self-supervised network, pruning makes big changes at the very beginning and then converges fast. But it requires a much longer "warm-up" phase that may involve complicated/entangled adjustments to many filter weights, if starting from a supervised network. Moreover, in practice, a new task might contain unseen classes uncovered by pre-training. While supervised networks may easily fail in this case and need re-training for the new classes, self-supervised networks are more robust since their nodes/filters capture semantic and more universal features not restricted to a closed set of classes.




(a) CIFAR-100, pruning ratio: 90% (b) Tiered-ImageNet, pruning ratio: 90%

Figure 3: IoU (mean±std) measuring filter sharing between two tasks of different similarity$\in \{0, 1, 2, 3, 4\}$ in each layer of their pruned models, for all the layers from input to output (left to right).

### 3.2 Do Pruned Models for More Similar Tasks share More Nodes/Filters in Every Layer?

The features learned for a task can still be helpful to similar tasks. This motivates various transfer/multi-task/meta learning methods, but does it still hold for pruning? In this study, we measure the similarity between two classification tasks in our dataset by the number of their shared classes (i.e., $|C_i \cap C_j|$ for task $i$ and $j$) and aim at finding whether/how it relates to their shared nodes/filters in different layers of their pruned models. In particular, let $\Omega_\ell^i$ and $\Omega_\ell^j$ denote the sets of filters remained in layer-$\ell$ after running Alg. 1 for task $i$ and $j$ (when using the same self-supervised network as the pre-trained model), we measure the overlap of the two sets by intersection over union (IoU) ratio (Jaccard, 1901), i.e., IoU $= |\Omega_\ell^i \cap \Omega_\ell^j| / |\Omega_\ell^i \cup \Omega_\ell^j|$. In Fig. 3, we show the IoU (mean±std) of each layer for pairs of tasks with task similarity$\in \{0, 1, 2, 3, 4\}$. For both datasets and pruning ratios, more similar tasks tend to share more filters (larger IoU) between their pruned models especially in the last few layers, because the features are more task-specific in deeper layers. Due to the same reason, though fluctuating between consecutive layers, IoU decreases with depth in the overall trend. Therefore, for a new task, its similar tasks' pruned models preserve many important filters and combining them might result in a better and much smaller initialization network.

### 3.3 Majority Voting among Similar Tasks Helps the Target Task's Pruning

The above empirical study about task similarity and shared filters between two tasks implies a simple strategy to initialize the pruned model for a new task, i.e., sampling filters according to their chances of being selected into the pruned models of similar tasks. In this empirical study, we aim at verifying this strategy on our dataset of pruned models and evaluating the recall of majority voting in MVP on selecting the crucial filters for target tasks.

In particular, we randomly select ten target tasks $A$ and for each target task-$i \in A$ we randomly select ten tasks $N_c^i$ of the same similarity $|C_i \cap C_j| = c, j \in N^i$. Among all filters in layer-$\ell$ that have been selected by $a$ tasks' models, i.e., $\bigcup_{j \in N_c^i} \Omega_\ell^j$, we identify the ones also preserved in target task-$i$'s model, i.e., $B_{i,\ell,c}(a) \triangleq \{k \in \bigcup_{j \in N_c^i} \Omega_\ell^j : |\{j \in N_c^i : k \in \Omega_\ell^j\}| = a\}$ and compute their

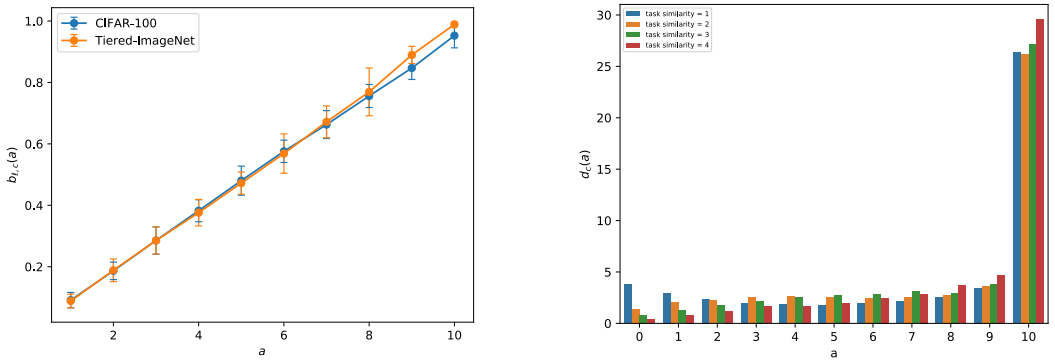



Figure 4: (**LEFT**) The odds $b_{\ell,c}(a)$ of a filter being selected in a target task's model if it is selected by $a$ out of 10 similar tasks: $a$ vs. $b_{\ell,c}(a)$ (mean±std) over all layers $\ell \in [L-1]$ and task similarities $c \in [4]$; (**RIGHT**) Histogram shows how many filters in the target task's model are selected by $a$ out of 10 similar tasks: $a$ vs. $d_{\ell,c}(a)$ (mean) over all $\ell \in [L-1]$. $b_{\ell,c}(a)$ and $d_{\ell,c}(a)$ are defined in Sec. 3.3.

proportion $b_{i,\ell,c}(a) \triangleq |B_{i,\ell,c}(a) \cap \Omega_\ell^i| / |B_{i,\ell,c}(a)|$. Averaging $b_{i,\ell,c}(a)$ over all target tasks $i \in A$, i.e., $b_{\ell,c}(a) \triangleq \sum_{i \in A} b_{i,\ell,c}(a) / |A|$, reflects the chance that a layer-$\ell$'s filter selected by $a$ similar tasks' models (similarity$= c$) is also selected by the target task's model. We plot $a$ vs. $b_{\ell,c}(a)$ for all layers $\ell \in [L-1]$ and $c \in [4]$ in Fig. 8 of Appendix. Since all the $(L-1) \times 4$ plots exhibit strong positive correlation between $a$ and $b_{\ell,c}(a)$, we summarize their mean and standard deviation in the left plot of Fig. 4. $b_{\ell,c}(a)$ is the recall of identifying the target task's filters by selecting the filters that receive $a$ votes from 10 similar tasks. It shows that the recall becomes higher if we select filters with more votes, i.e., a filter selected by more similar tasks will have higher chance of being preserved in the pruned model for the target task, which indicates the effectiveness of majority voting.

On the other hand, does every filter in the target task-$i$'s pruned model has a good chance of being selected by the similar tasks' models? We calculate the number of filters in layer-$\ell$ of target task-$i$'s pruned model that have been selected by $a$ tasks in $N^i$, i.e., $d_{i,\ell,c}(a) \triangleq \left| \left( [n_\ell] \setminus \bigcup_{j \in N_c^i} \Omega_\ell^j \right) \cap \Omega_\ell^i \right|$ for $a = 0$ and $d_{i,\ell,c}(a) \triangleq |B_{i,\ell,c}(a) \cap \Omega_\ell^i|$ for $a \geq 1$. We report the mean $d_c(a)$ of $d_{i,\ell,c}(a)$ over all target tasks $i \in A$ for different $(\ell, c)$ by $4(L-1)$ histograms in Fig. 9 of Appendix and summarize their averaged histogram over all layer-$\ell \in [L-1]$ in the right plot of Fig. 4. This plot evaluates the recall under different task similarities: it shows how the numerator of the recall, $d_c(a)$ (i.e., the number of filters shared between the target task and training tasks similar to it), changes with the task similarity and the votes. It implies that selecting filters with more votes from more similar tasks (red bars for $a = 8, 9, 10$ ) improves the recall. Moreover, using tasks with higher similarity can significantly reduce the uncovered filters ($a = 0$).

## 4 Meta-Vote Pruning (MVP)

Inspired by the empirical study above, we propose a simple yet strong baseline "meta-vote pruning (MVP)" for non-parametric meta-pruning. Given a target task $i$, MVP draws a sub-network of a self-supervised network by sampling filters in each layer using majority voting among the pruned models of similar tasks $N^i$. In particular, we apply softmax (with temperature $\tau$) to the number of tasks in $N^i$ that select each filter-$k \in [n_\ell]$ from layer-$\ell$ of the pre-trained model, which yields a probability distribution over all the filter $[n_\ell]$, i.e., $\forall k \in [n_\ell]$,

$$p(k) = \frac{\exp(|\{j \in N^i : k \in \Omega_\ell^j\}|/\tau)}{\sum_{h \in [n_\ell]} \exp(|\{j \in N^i : h \in \Omega_\ell^j\}|/\tau)} \tag{1}$$

To initialize layer-$\ell$ of the sub-network, MVP samples filters from this distribution (without replacement) according to the targeted pruning ratio $r$. We further initialize the parameters of each filter-$k$ by averaging its parameters in the pruned models of the similar tasks $\{j \in N^i : k \in \Omega_\ell^j\}$. MVP then fine-tunes the initialized sub-network for a few iterations on the training set of the target task.

## 5 Experiments

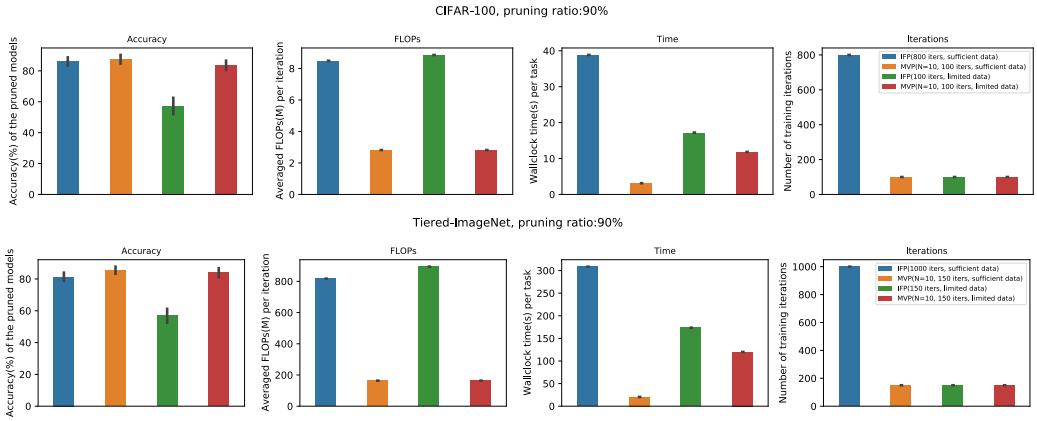



Figure 5: Test accuracy, memory and computational costs of MVP and IFP in the experiments.

In this section, we conduct three groups of experiments on two datasets, i.e., CIFAR-100 (Krizhevsky, 2009) and Tiered-ImageNet (Ren et al., 2018), which evaluate MVP (Alg. 2) and compare it mainly with IFP (Alg. 1) under different number of iterations (**Group-I**, top plot, hundreds vs. thousands iterations), different amount of training data (**Group-II**, middle plot, 500(1300) vs. 100(200) training samples per class on CIFAR-100(Tiered-ImageNet)), task similarities for general cases without overlapping classes between tasks (**Group III**, bottom plot) in Fig. 6. We further compare their computational and memory efficiency in terms of iterations, wall-clock time, and FLOPs in Fig. 5. The bars in the accuracy plot of Fig. 5 are representative results copied from Fig. 6. MVP outperforms IFP on the test-task accuracy by a large margin especially in the region of fewer iterations, limited training data, or/and different task similarity. In some experiments, MVP even outperforms IFP that uses much more iterations or data. Meanwhile, it saves a great amount of memory and computation and thus creates much less carbon footprint than IFP. In addition, the simple majority voting in MVP is powerful in extracting helpful meta-information for test tasks' pruning even from less similar training tasks (e.g., sharing only one class or parent class).

**Algorithm 2** META-VOTE PRUNING (MVP)

**Input** : Target task $i$ and its training set $D_i$, pruning ratio $r$, $\tau$, $J$, $N$, a dataset of pruned models for different tasks
**Output** : A pruned model for target task-$i$
**Initialize** : $\Omega_\ell \leftarrow \emptyset$, the set of filters in layer-$\ell$
Sample/find $N$ similar tasks $N^i$ to task $i$;
**for** $\ell \leftarrow 1$ ***to*** $L-1$ **do**
  Sample $(1-r)n_\ell$ filters with probability $p(k)$ (Eq. (1)) and add them to $\Omega_\ell$;
  **for** $k \in \Omega_\ell$ **do**
    Initialize filter-$k$ by averaging its parameters for tasks in $\{j \in N^i : k \in \Omega_\ell^j\}$;
  **end**
**end**
Fine-tune the pruned model for $J$ iterations on $D_i$.

## 5.1 IMPLEMENTATION DETAILS

In every group of experiments, we randomly draw 30 test tasks from the dataset introduced in Sec. 3 and treat the rest 280 tasks as training tasks. For every test task (i.e., the target task in Alg. 2), we apply MVP on multiple sets of training tasks with identical task similarity to the test task: (1) for Group-I&II, the task similarity is measured by the shared classes $c = |C_i \cap C_j|$ and we tried four sets of training tasks with $c \in \{1, 2, 3, 4\}$; (2) for Group-III, we use a similarity metric based on the class prototypes. In particular, we first compute its prototype for every class of each task by averaging its samples' penultimate-layer representations of the pre-trained model. To compute the similarity between two tasks, we apply bipartite graph matching (Hungarian algorithm (Kuhn, 1955)) to the two sets of class prototypes. The edge weight is defined by the cosine similarity between two class prototypes. The task similarity is defined as the averaged cosine similarity among all matched pairs of classes from the two tasks. Each method (MVP, IFP, uniform pruning) in every group of experiments is represented by one bar in Fig. 6, which reports the mean and standard deviation of test-set accuracy achieved by the pruned model over all 30 test tasks.

In Fig. 6, we report the performance of MVP using different combinations of hyperparameters $N$ (number of similar tasks), $J$ (iterations), and $r$ (pruning ratio). We tune temperature $\tau$ on ten validation experiments and choose $\tau = 0.025$ for all experiments. For comparison, we also report

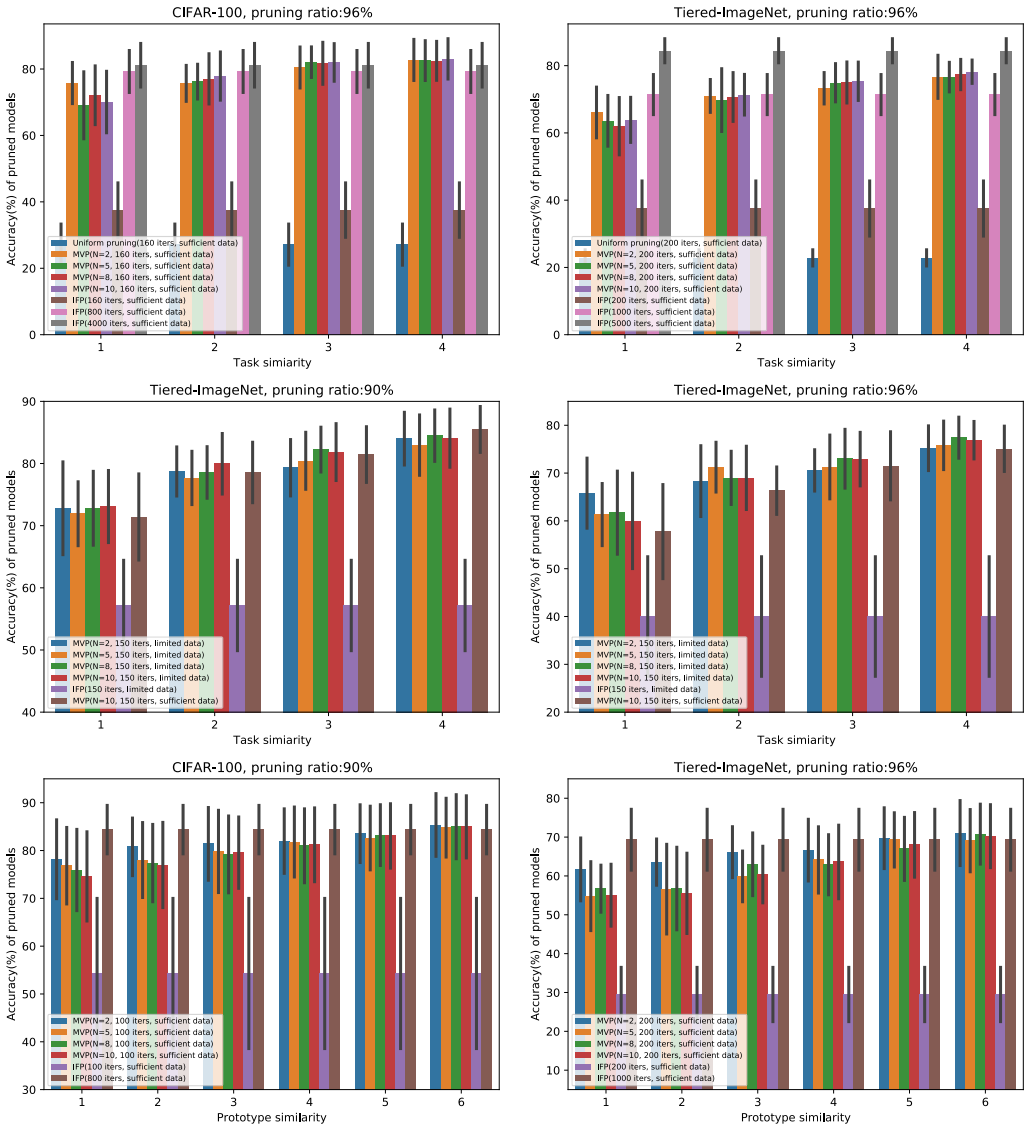



Figure 6: Test-set accuracy of pruned models produced by different methods (MVP, IFP, uniform pruning) using different number of iterations ( **Group-I, Top plots**), different amount of training data ( **Group-II, Middle plots**), and training tasks with prototype similarity ( **Group-III, Bottom plots**).

the performance of IFP and uniform pruning in different settings. IFP does not use any meta knowledge from the pruned models of training tasks and comparison to it can reveal the advantages of meta-pruning in different scenarios. Uniform pruning replaces the majority voting (line 3 in Alg. 2) with uniform sampling of filters from the pre-trained self-supervised network, which is the "random baseline" for majority voting. We use the same optimizer for all methods, i.e., SGD with momentum and cosine-annealing learning rate schedule. We use a learning rate of 0.0003(0.0005) on CIFAR-100 (Tiered-ImageNet) in MVP and uniform pruning. We tune the learning rate of IFP for different amount of training data due to the sensitivity of large-model fine-tuning to training data size. Specifically, we set it to 0.001 for limited data and 0.0001 for sufficient data.

### 5.2 Detailed Analysis

**Group-I** In the top-row plot of Fig. 6, we compare MVP using fewer iterations, uniform pruning with fewer iterations, and IFP using both fewer and much more iterations, when using all the available training data. It shows that the uniform pruning performs much poorer than MVP without using the majority voting information to select filters, which implies that the pruned models of similar tasks can significantly improve the quality of initialized network for a new task. MVP's performance

improves as it is able to access more similar training tasks to the test task or/and select more of them (increasing $N$). When using low-similarity training tasks, the performance of MVP is still acceptable and comparable to other baselines, but increasing $N$ does not always bring improvement due to the noises from unshared classes. On both datasets, MVP outperforms IFP who spends $5\times$ iterations when using training tasks of similarity 3 or 4. On CIFAR-100, it even outperforms IFP that costs $25\times$ iterations. Hence, MVP can produce higher-quality pruned model when using fewer iterations.

**Group-II** In the middle-row plot of Fig. 6, we compare MVP with IFP when training data is limited to a small amount and the pruning iterations are limited to hundreds. MVP exhibits overwhelming advantages over IFP in this region. Its accuracy improves with the increasing similarity, which is consistent with results in Group-I. In addition, we witness that MVP with limited data has comparable accuracy as MVP with sufficient ($\geq 5\times$) data when the number of iterations are limited. This is different from observations on IFP, whose performance heavily depends on the availability of sufficient data. Therefore, the meta knowledge MVP extracting from similar training tasks can make the pruning much less data-hungry, which is preferred in the applications that aim to deploy a large pre-trained model to edge devices with limited local data.

**Group-III** In the bottom-row plot of Fig. 6, we evaluate the effectiveness of MVP using a more general similarity metric (prototype similarity) for the case that there are no overlapping classes between different tasks. In this experiment, classes of the test tasks will never appear in any training task, i.e., no overlapping class between any training and test tasks. For a better comparison, we quantize the prototype similarity to 6 levels as the shared class similarity in Fig. 6. We observe a degradation of the accuracy comparing to the previous "shared classes similarity" case since the set of classes of these tasks is disjoint and these tasks are less similar. However, the protptype similarity still lends MVP the power to surpass IFP and approximate IFP with much more iterations, which again verify the importance of meta knowledge in finding a better initialization sub-network. Moreover, we still can see an improvement of accuracy when increasing the prototype similarity.

**Efficiency** In Fig. 5, we compare the memory and computational cost of MVP with IFP on two datasets when targeting two pruning ratios and using two different amounts of training data. All the presented experiments are accomplished on a 24GB RTX 6000 GPU. In all the 8 scenarios, MVP significantly reduces the FLOPs and computational iterations/time required by IFP while achieves higher test accuracy. The improvement of MVP on the efficiency is attributed to two factors, i.e., the few iterations it requires to find a high-quality pruned model, and a much smaller (yet high-quality) sub-network to initialize the weight fine-tuning.

## 6 Conclusion

In this paper, we study "non-parametric meta-pruning" problem that aims to reduce the memory and computational costs of single-task pruning, via reusing a pre-trained model and similar tasks' pruned models to find an initialization sub-network for a new task. We conduct three empirical studies to investigate (1) the choices of pre-trained model for meta-pruning; (2) the relationship between task similarity and the pruned models of two tasks; and (3) the proximity of a sub-network selected by majority voting among similar tasks to the ground-truth pruned model of the target task. The first study reveals several advantages of adopting self-supervised network as the pre-trained model for meta-pruning, while the rest two studies motivates a simple yet strong baseline for meta-pruning, called "meta-vote pruning (MVP)" (Alg. 2). By extensive experiments on multiple tasks drawn from two datasets under different training settings, we demonstrate the advantages of MVP over other pruning methods in the region of limited computation or/and data and show its potential on reducing carbon footprint of pruning/fine-tuning large networks for billions of edge devices and tasks.

## References

Mathilde Caron, Ari S. Morcos, Piotr Bojanowski, J. Mairal, and Armand Joulin. Pruning convolutional neural networks with self-supervision. *ArXiv*, abs/2001.03554, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. *ArXiv*, abs/2006.10029, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Ting-Wu Chin, Ruizhou Ding, C. Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1515–1525, 2020.

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*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *ArXiv*, abs/1708.04552, 2017.

Chelsea Finn, P. Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL http://arxiv.org/abs/1803.03635.

Jean-Bastien Grill, Florian Strub, Florent Altch'e, C. Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, B. A. Pires, Zhaohan Daniel Guo, M. G. Azar, Bilal Piot, K. Kavukcuoglu, R. Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733, 2020.

Song Han, Jeff Pool, John Tran, and W. Dally. Learning both weights and connections for efficient neural network. *ArXiv*, abs/1506.02626, 2015.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

Kaiming He, X. 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, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Yang He, Ping Liu, Linchao Zhu, and Y. Yang. Meta filter pruning to accelerate deep convolutional neural networks. *ArXiv*, abs/1904.03961, 2019.

Geoffrey E. Hinton, Oriol Vinyals, and J. Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Paul Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 01 1901. doi: 10.5169/seals-266450.

Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and H. Pirsiavash. Compress: Self-supervised learning by compressing representations. *ArXiv*, abs/2010.14713, 2020.

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

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

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Hao Li, Asim Kadav, Igor Durdanovic, H. Samet, and H. Graf. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2017.

Yawei Li, Shuhang Gu, K. Zhang, L. Gool, and R. Timofte. Dhp: Differentiable meta pruning via hypernetworks. *ArXiv*, abs/2003.13683, 2020.

Shaohui Lin, R. Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and B. Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, 2018.

Z. Liu, Haoyuan Mu, X. Zhang, Zichao Guo, X. Yang, K. Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3295–3304, 2019.

Zhuang Liu, J. Li, Zhiqiang Shen, Gao Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2755–2763, 2017.

Christos Louizos, M. Welling, and Diederik P. Kingma. Learning sparse neural networks through l0 regularization. *ArXiv*, abs/1712.01312, 2018.

Eran Malach, Gilad Yehudai, S. Shalev-Shwartz, and O. Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *ICML*, 2020.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *ArXiv*, abs/1803.02999, 2018.

Mengye Ren, Eleni Triantafillou, S. Ravi, J. Snell, Kevin Swersky, J. Tenenbaum, H. Larochelle, and R. Zemel. Meta-learning for semi-supervised few-shot classification. *ArXiv*, abs/1803.00676, 2018.

Pedro H. P. Savarese, Hugo Silva, and M. Maire. Winning the lottery with continuous sparsification. *ArXiv*, abs/1912.04427, 2020.

Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and X. Huang. Learning sparse sharing architectures for multiple tasks. *ArXiv*, abs/1911.05034, 2020.

Chong Min John Tan and Mehul Motani. DropNet: Reducing neural network complexity via iterative pruning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9356–9366. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/tan20a.html.

Hongduan Tian, Bo Liu, X. Yuan, and Qingshan Liu. Meta-learning with network pruning. *ArXiv*, abs/2007.03219, 2020.

Mao Ye, L. Wu, and Qiang Liu. Greedy optimization provably wins the lottery: Logarithmic number of winning tickets is enough. *ArXiv*, abs/2010.15969, 2020.

J. Zbontar, L. Jing, Ishan Misra, Y. LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *ArXiv*, abs/2103.03230, 2021.

Tao Zhuang, Zhixuan Zhang, Yuheng Huang, X. Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. In *NeurIPS*, 2020.