Vote for Nearest Neighbors Meta-Pruning of Self-Supervised Networks

Anonymous Authors[1]

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? 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 of 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 and 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 and nodes. Numerous empirical results have verified that pruning can compress the original network to much smaller sub-networks that still enjoy comparable performance.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022. Do not distribute.

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 and waste a lot of computational power. 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. Meta-pruning follows a 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., 2020; 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. We further discover a strong correlation between the chance that a filter is retained after the pruning for a task and the number of times it is also preserved in pruned models for similar tasks.

Motivated by above empirical study, 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 filters through majority voting among similar tasks, i.e., we sample nodes/filters according to their chances being selected into the pruned model of similar tasks. To keep this baseline simple, we sample the same proportion of 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. The proposed baseline saves a substantial amount of computation and memory while still maintains a high test accuracy of the pruned models. 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. 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. There are two major choices for the pre-trained model: (1) a neural network trained by supervised learning on a labeled dataset, e.g., ImageNet; (2) a neural network trained by self-supervised learning on unlabeled data that are widely available. 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), respectively.

Pruning Algorithm Iterative pruning alternates between network pruning and fine-tuning of model weights for multiple iterations usually achieves better performance in empirical comparisons with other pruning methods. This empirical study is conducted on convolutional neural networks so we apply iterative filter-pruning (IFP) that removes filters with the smallest activation values averaged over all training samples. 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-ℓ composed of $\theta_{\ell,i}$ for every filter-i. IFP first train a new linear classifier (i.e., θ_L) for K epochs on D_T and follows by J fine-tune iterations. It prunes $p\%$ of the filters remained in each layer every h iterations according to their activation values $f_{\ell,i}(x)$. It stops to prune layer-ℓ if reaching the targeted pruning ratio r. Refer to appendix for detailed algorithm.

A Dataset of Pruned Models Our empirical study is carried out on CIFAR-100 and Tiered-ImageNet. For each dataset, we randomly draw 300 classification tasks, each defined on 5 classes sampled without replacement. Running IFP for all 300 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 θ_T. We use the same hyper-parameters for different tasks. As argued in Sec. 1, limited iterations and small learning rate are applied to compare the pre-trained models (Sec. 2.1), 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). We have tried two pruning ratios $r = \{0.9, 0.96\}$. To study the similarity between the "winning tickets" for different tasks, we use more delicately pruned models achieved by running more iterations ($J = 800(1000)$) in above experiments.

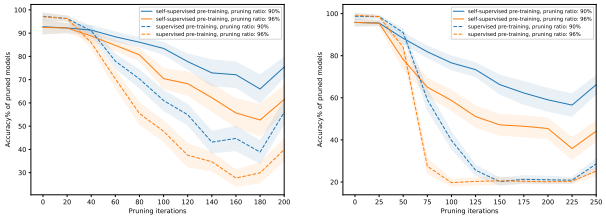


Figure 1. Supervised vs. self-supervised network for IFP with fewer iterations: test accuracy (mean±std) of pruned models with 200 and 250 pruning iterations for 300 tasks drawn from CIFAR-100 (**LEFT**) and Tiered-ImageNet (**RIGHT**).

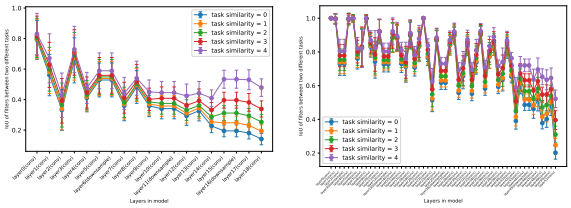
2.1. Which Pre-trained Model to Prune? Supervised vs. Self-Supervised Model

We compare supervised and self-supervised networks when pruned for 300 different tasks drawn from CIFAR-100 (Tiered-ImageNet) using IFP 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. 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. So pruning a self-supervised network for a

new task can be easier and starts closer to the final model.

2.2. Do Pruned Models for More Similar Tasks share More Nodes/Filters in Every Layer?

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 how it relates to their shared filters in different layers of their pruned models. In particular, let Ω_ℓ^i and Ω_ℓ^j denote the sets of filters remained in layer-ℓ after running IFP for task i and j, we measure the overlap of the two sets by intersection over union (IoU) ratio (Jaccard, 1901), i.e., $\text{IoU} = |\Omega_\ell^i \cap \Omega_\ell^j| / |\Omega_\ell^i \cup \Omega_\ell^j|$. In Fig. 2, we show the IoU (mean$\pm$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. 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.



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

Figure 2. IoU (mean$\pm$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. 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 τ) to the number of tasks in N^i that select each filter-$k \in [n_\ell]$ from layer-ℓ of the pre-trained model, which yields a probability distribution over all the filter $[n_\ell]$, i.e., $\forall k \in [n_\ell]$,

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

To initialize layer-ℓ 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. Refer to appendix for detailed algorithm.

4. Experiments

In this section, we conduct three groups of experiments to evaluate MVP and compare it mainly with IFP under different number of iterations, different amount of training data and prototype similarities for general cases without overlapping classes between tasks in Fig. 3. In Table 1, we compare MVP with SOTA pruning methods. In appendix, we further compare their computational and memory efficiency in terms of iterations, wall-clock time, and FLOPs in Fig. 4. And the results of applying MVP to unseen datasets are reported in Table 2 in appendix.

4.1. Implementation Details

In every group of experiments, we randomly draw 30 test tasks from the dataset introduced in Sec. 2 and treat the rest 270 tasks as training tasks. For every test task, 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 so that classes in different tasks are disjoint. In particular, we first compute its prototype for every class of each task and compute the similarity between two tasks by applying bipartite graph matching (Hungarian algorithm (Kuhn, 1955)) to the two sets of class prototypes.

In Fig. 3, we report the performance of MVP. For comparison, we also report 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 uniform pruning replaces the majority voting 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.

4.2. Detailed Analysis

Group-I In the left plot of Fig. 3, we mainly compare different methods with fewer iterations. It shows that the uniform pruning performs much poorer than MVP without meta knowledge, which implies that the pruned models of

Table 1. Comparison with SoTA methods. Number in brackets refers to training iterations.

	MEST+EM(800)	MEST+EM&S(800)	MEST+EM(100)	MEST+EM&S(100)	DLTH(800)	HT-based Reptile(100)	MVP(100)
Accuracy	78.84 ± 5.37	79.12 ± 5.13	75.78 ± 7.27	76.28 ± 6.84	77.4 ± 6.73	73.65 ± 2.58	87.08 ± 4.39

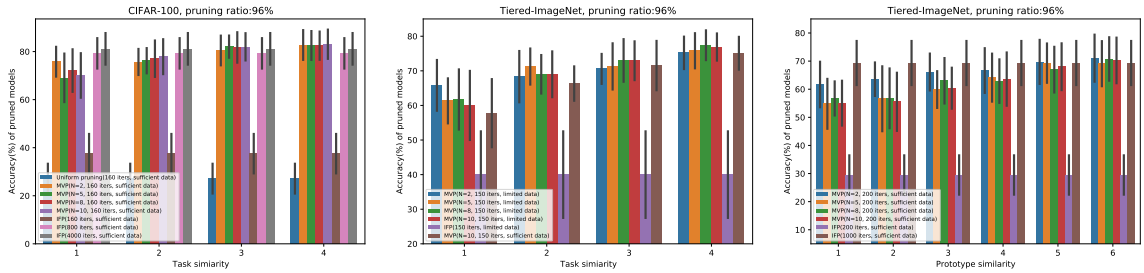


Figure 3. Test-set accuracy of pruned models produced by different methods (MVP, IFP, uniform pruning) using different number of iterations (**Group-I, Left plot**), different amount of training data (**Group-II, Middle plot**), and training tasks with prototype similarity (**Group-III, Right plot**).

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. MVP outperforms IFP who spends $5\times$ iterations when using training tasks of similarity 3 or 4. 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 plot of Fig. 3, we compare MVP with IFP when training data and pruning iterations are limited. 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.

Group-III In the right plot of Fig. 3, 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. For a better comparison, we quantize the prototype similarity to 6 levels. 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 prototype 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.

Comparison to SoTA methods In Table 1, we compare MVP with SOTA related methods on the tasks of CIFAR-100. We compare MVP with IHT-based Reptile (Tian et al., 2020), which is a meta-pruning method that applies meta-learning to initializing the model weights for pruning. In particular, IHT-based Reptile applies Reptile (Nichol et al., 2018) and iterative pruning to find better weight-initialization for a pruned meta-model. The results show that MVP achieves higher accuracy than IHT-based Reptile when their training iterations and the amount of training data are the same, implying that MVP can find better initialization of pruned models for different tasks. MEST (Yuan et al., 2021) is the SOTA method in sparse training community of which the training starts from a sub-network. The difference is that MVP is trained on the sub-network with meta knowledge while the sub-network for MEST is randomly initialized. DLTH (Bai et al., 2022) is a variant of Lottery Ticket Hypothesis which propose a method to transform random tickets into winning tickets, but the winning tickets generated by DLTH is worse than MVP. In Table 1, MVP outperforms these methods by a large margin, which implies meta knowledge from similar tasks can speed up training of target task and promote the final accuracy.

5. 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 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. The first study reveals several advantages of adopting self-supervised network as the pre-trained model for meta-pruning, while another motivates a simple yet strong baseline for meta-pruning, called "meta-vote pruning (MVP)". 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

Bai, Y., Wang, H., Tao, Z., Li, K., and Fu, Y. Dual lottery ticket hypothesis. *arXiv preprint arXiv:2203.04248*, 2022.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020.

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

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

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

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

Grill, J.-B., Strub, F., Altch'e, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733, 2020.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Jaccard, P. 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.

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

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

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

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *ArXiv*, abs/1803.02999, 2018.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

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

Tian, H., Liu, B., Yuan, X., and Liu, Q. Meta-learning with network pruning. *ArXiv*, abs/2007.03219, 2020.

Yuan, G., Ma, X., Niu, W., Li, Z., Kong, Z., Liu, N., Gong, Y., Zhan, Z., He, C., Jin, Q., et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34, 2021.

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

A. Iterative filter pruning algorithm

Algorithm 1 ITERATIVE FILTER PRUNING (IFP)

Input : Pre-trained network $F(\cdot;\theta)$, Task T and training set D_T, K, Hyper-parameters J, h, r, p
Initialize : $\Omega_\ell \leftarrow [n_\ell]$, the set of filters preserved in layer-ℓ

Replace the last fully-connected layer θ_L of $F(\cdot;\theta)$ with a newly initialized one for targeted task T;
Train θ_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 Ω_ℓ with 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 θ_L;
end
Fine-tune the pruned model for one epoch on D_T;

B. Meta-vote pruning algorithm

Algorithm 2 META-VOTE PRUNING (MVP)

Input : Target task i and its training set D_i, pruning ratio r, τ, 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-ℓ

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 Ω_ℓ;
  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.

C. Comparison of efficiency between different methods.

In Fig. 4, the comparison between the memory and computational cost of MVP, IHT-based Reptile and 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 scenarios, MVP is as efficient as IHT-based Reptile and 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.

D. Application of MVP to unseen datasets

In Table 2, the results of testing MVP on unseen dataset Oxford Flowers (Nilsback & Zisserman, 2008) based on the Tiered-ImageNet pruned models using prototype similarity are reported. Results are consistent with our observations that MVP outperforms IFP with the same limited iterations, and tasks with higher similarities leads to better performance. This result indicates that MVP can be applied to independent datasets and achieve good performance.

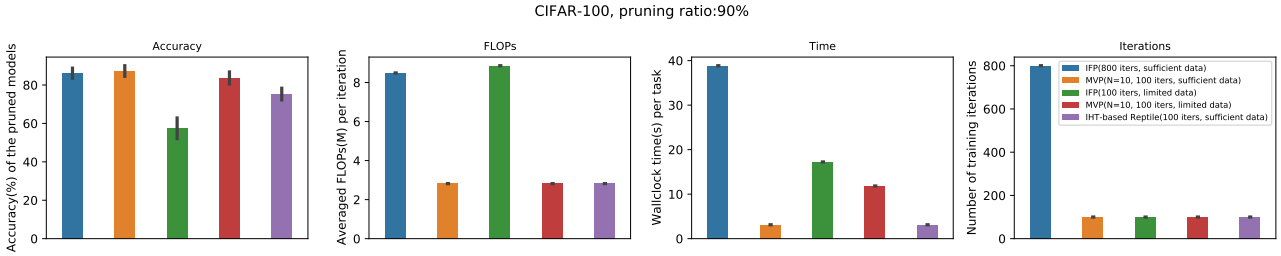


Figure 4. Test accuracy, memory and computational costs of MVP, IFP and IHT-based Reptile (Tian et al., 2020).

Table 2. Accuracy of testing MVP on independent Oxford Flowers dataset

Methods\Similarity	1	2	3	4	5	6
MVP(N=2,200 iters)	80.40±10.65	81.40±8.77	81.20±10.81	79.00±8.40	84.00±7.48	83.20±7.76
MVP(N=5,200 iters)	79.40±9.76	76.60±7.54	81.80±8.78	77.80±6.42	79.00±10.13	80.00±7.59
MVP(N=8,200 iters)	75.00±8.77	74.80±11.97	74.00±9.84	80.80±10.85	77.80±11.78	79.80±8.17
MVP(N=10,200 iters)	72.40±13.08	76.60±11.07	79.40±6.33	74.00±10.77	80.60±8.30	78.20±13.70
IFP(200 iters)	66.40±13.11	66.40±13.11	66.40±13.11	66.40±13.11	66.40±13.11	66.40±13.11