VOTING FROM NEAREST TASKS: META-VOTE PRUNING OF PRE-TRAINED MODELS FOR DOWNSTREAM TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

As a few large-scale pre-trained models become the major choices of various applications, new challenges arise for model pruning, e.g., can we avoid pruning the same model from scratch for every downstream task? How to reuse the pruning results of previous tasks to accelerate the pruning for a new task? To address these challenges, we create a small model for a new task from the pruned models of similar tasks. We show that a few fine-tuning steps on this model suffice to produce a promising pruned-model for the new task. We study this "meta-pruning" from nearest tasks on two major classes of pre-trained models, convolutional neural network (CNN) and vision transformer (ViT), under a limited budget of pruning iterations. Our study begins by investigating the overlap of pruned models for similar tasks and how the overlap changes over different layers and blocks. Inspired by these discoveries, we develop a simple but effective "Meta-Vote Pruning (MVP)" method that significantly reduces the pruning iterations for a new task by initializing a sub-network from the pruned models of its nearest tasks. In experiments, we demonstrate MVP's advantages in accuracy, efficiency, and generalization through extensive empirical studies and comparisons with popular pruning methods over several datasets.

1 INTRODUCTION

Large-scale pre-trained models usually contain tens of millions or even billions of parameters for promising generalization performance. The computation and memory of modern GPUs or clusters can support to train such 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 and nodes. Numerous empirical results have verified that pruning can compress the original network into much smaller sub-networks that still enjoy the comparable 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 for a wide variety of devices and adapted to different downstream tasks. Running an iterative pruning algorithm for every device or task from the same pre-trained network can create enormous carbon footprint overload in our biosphere and waste a lot of computational power. On the other hand, the wide applications of a few pre-trained models have already created thousands of pruned models for different downstream tasks. Can we reuse these pruned models as prior knowledge to save the pruning computation on new tasks? We call this problem "meta-pruning". In this paper, we mainly focus on a special case of it, which initializes a sub-network for a given new task based on the pruned models of similar 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 of a network. It differs from MAML in that (1) both the sub-network's architecture and weights are initialized; and (2) the initialization is not universal but task-specific.

Since meta-pruning aims to find better sub-network initialization for new tasks, we limit the iterations during meta-pruning to strengthen the impact of initialization on the final pruned model. This also controls the computational cost and carbon footprint of meta-pruning much less than conventional pruning that requires many iterations. Under this constraint, a well-performed pre-trained model is critical to the meta-pruning performance because (1) it needs to provide initialized sub-networks for different tasks; and (2) a few iterations of fine-tuning to the sub-networks should suffice to produce high-quality pruned models for targeted tasks. Meta-pruning follows a practical setting where one single pre-trained model is tailored for different tasks using limited iterations. We study two classes of the most widely used pre-trained models, i.e., convolutional neural networks (CNN) (He et al., 2016) and vision transformer (ViT) (Dosovitskiy et al., 2021).

The primary contribution of this paper is two folds. In the first part, we conduct a thorough empirical study that applies different pruning methods to CNN and ViT and compare their produced sub-networks for hundreds of downstream tasks. No meta-pruning is studied in this part and its primary purpose is to (1) find the nearest tasks for a new task using different similarity metrics; and (2) compare the pruned models for different but similar tasks. To this end, we build a dataset of tasks and their sub-networks pruned from the same pre-trained models. Statistics and evaluations on this dataset indicate similar tasks with high similarity tend to share more nodes/filters/heads preserved in their pruned models, especially in deeper layers that notably capture high-level task-specific features.

Motivated by the empirical study, the second part of this paper proposes a simple yet strong meta-pruning method called "meta-vote pruning (MVP)". It can significantly reduce the pruning cost and memory required by previous pruning approaches yet still produce pruned models with promising performance. Given a pre-trained model, MVP finds a sub-network for a new task by selecting nodes/filters/heads through majority voting among its nearest tasks, e.g., a filter will be sampled with a higher chance if it is selected into more sub-networks of similar tasks. To keep the method simple, we sample the same proportion of nodes/filters/heads as the targeted pruning ratio. Then we apply a few iterations of fine-tuning to the initialized sub-network using training data of the new task. Although a more sophisticated procedure can be developed, the proposed method already saves a substantial amount of computation and memory while maintaining a high test accuracy of pruned models. We demonstrate these via experiments over tasks from CIFAR-100 (Krizhevsky & Hinton, 2009), ImageNet (Deng et al., 2009) and Caltech-256 (Griffin et al., 2007). The pruned models extracted from an ImageNet pre-trained model can also vote for tasks drawn from the unseen dataset Caltech-256 with great performance, which shows the generalization of MVP.

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 structure pruning below. In CNN, 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 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. Methods (Bai et al., 2022; Frankle & Carbin, 2018) based on lottery ticket hypothesis try to find a well-performed sparse initialization for each task.

ViT has been widely used in computer vision and achieved SOTA performance in many tasks. The input patches for each block can be pruned to save computation for ViT. Goyal et al. (2020) propose a metric for the importance of each patch and dynamically prune patches in each layer. PatchSlimming (Tang et al., 2022) retains patches critical to preserve the original final output. HVT (Pan et al., 2021) is a CNN-like method which shortens the patch sequence by max-pooling. Another line of work (Zhu et al., 2021; Yu et al., 2022b;a) automatically prunes the unimportant heads, nodes and blocks in ViT. These methods excel on single-task pruning but their cost linearly increases for multiple tasks (and thus more expensive than meta-pruning) because: (1) a large model needs to be trained for every task; (2) every task requires to prune its own large pre-trained model from scratch. For both CNN and ViT, it is time-consuming for these pruning methods to build a pruned model for each unseen target task from a large pre-trained model. Our proposed method can borrow

the knowledge of the existing pruned models extracted by these pruning methods and use them to generate a well-performed pruned model for the unseen task with a few fine-tuning iterations.

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 other 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-learning has also been studied to select the best pruning criterion for different tasks (He et al., 2019). In (Sun et al., 2020), a shared sparse backbone network is trained for multi-task learning but it cannot be adapted to new tasks. Our method is the first one to use meta-learning to extract a pruned model for a new task. The main differences of our approach to them are: (1) we do not train a parametric meta-learner but instead use majority voting from similar tasks; and (2) our meta-voting generates a pruned small sub-network to initialize the target task training, which significantly reduces the pruning cost.

3 Empirical Study: Pruning a Pre-trained model for Different Tasks

In this section, we conduct an empirical study that applies different methods to prune a CNN or ViT pre-trained model for over hundreds of tasks. Our study focuses on the overlap between the pruned models for different tasks and whether/how it relates to their similarity. To this end, we introduce different task similarities and compare the overlap associated with different similarity groups. The results show that more similar tasks tend to share more nodes/filters/heads in their pruned models. And this holds across different pruning methods, datasets and pre-trained models. No meta-pruning is used in the study.

3.1 A Dataset of Pruned Models

While the number of possible downstream tasks and users can be huge in practice, the current progress on foundation models show that one or a few large-scale pre-trained models with light fine-tuning usually achieve the SOTA performance on most of them. To simulate this scenario on a standard dataset, our empirical study creates a dataset of pruned models for hundreds of tasks from the same pre-trained model. We choose CIFAR-100 (Krizhevsky & Hinton, 2009) and ImageNet (Deng et al., 2009) for the study due to many classes in them. Each class in CIFAR-100 and ImageNet has 500 and 1300 samples, respectively. For each dataset, we randomly draw 1000 classification tasks, each defined on 5 classes sampled without replacement. We adopt ResNet-18 (He et al., 2016) pre-trained on CIFAR-100, ResNet-50 (He et al., 2016) and a small version of ViT (Touvron et al., 2021b) pre-trained on ImageNet. For ResNet-18 and ResNet-50, we prune two types of pre-trained models, i.e., the supervised training following Devries & Taylor (2017) and the self-supervised training following SimSiam Chen & He (2020) (only the encoder is used). For ViT, the training of its pre-trained model follows (Dosovitskiy et al., 2021).

Iterative Pruning We apply iterative filter-pruning (IFP) to ResNet. Unlike magnitude-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 progressively prunes a large network to the targeted size. It usually performs better than other pruning methods and has also been mainly studied in theoretical works about Lottery Ticket Hypothesis (Frankle & Carbin, 2018). We take the activation values of filters averaged over all training samples to measure the importance of filters (Molchanov et al., 2016), referred as Activation Pruning, in which filters with smaller activation values contain less information of input data. The detailed procedure of IFP is described in Alg. 2 in Appendix.

Automatic Pruning Inspired by the SOTA ViT structured pruning method (Yu et al., 2022b), we prune ViT by automatic head&node pruning (AHNP) for a given task, which parameterizes the sub-

network as the pre-trained model with a learnable score multiplied to each prunable head and node. To encourage sparsity, the differentiable scores of all prunable heads and nodes are optimized with an additional $L1$ regularization loss. After each optimization step, we apply a simple thresholding to these scores to remove heads and nodes with small scores. The optimization stops if the pruned model reaches the targeted size and the model will be fine-tuned for a few iterations. The detailed procedure of AHNP can be found in Alg. 3 in Appendix. For tasks of CIFAR-100, we run IFP for all 1000 tasks on ResNet-18. And we apply IFP and AHNP to tasks of ImageNet on ResNet-50 and ViT respectively. Finally, we create a dataset of pruned models for thousands of tasks over different pre-trained models. For each task i, we record its labels C_i, the set of preserved nodes/filters/heads $\{\Omega_\ell\}_{\ell=1:L-1}$ and the pruned model θ_T. We use the same hyper-parameters for different tasks. For IFP on ResNet, we use a learning rate of 0.005, pruning iterations of 1000 and batch-size of 128 for both the tasks of CIFAR-100 and ImageNet. When applying AHNP to ViT, we follow the ViT training in (Touvron et al., 2021a). We reduce the pruning iterations to 1000 and use a small learning rate of 0.00005 for parameters inherited from the pre-trained ViT (to preserve its knowledge) and a large learning rate of 0.05 for the learnable scores. The pruning ratio is 0.9 for all pruned models.

3.2 Do similar tasks share more nodes/filters/heads on each layer of their pruned models?

The representations learned for a task can still be helpful to its similar tasks. This motivates transfer/multi-task/meta learning methods. But do similar tasks also share more structures in their pruned sub-networks? We apply two metrics to measure the similarity between classification tasks in our dataset and study whether/how the similarity relate to their shared nodes/filters/heads in different layers of their pruned models.

Similarity Metrics We apply two metrics to compute the similarity between tasks and find the nearest tasks, i.e., the Log Expected Empirical Prediction (LEEP) (Nguyen et al., 2020) and the Wordnet wup similarity (Pedersen et al., 2004; Wu & Palmer, 1994). LEEP score is widely used in transfer learning to estimate the knowledge transferability from a source task to a target task. In our study, for each target task, we can rank the other tasks by their LEEP similarity score from each of them to the target one. Computing the LEEP score only requires a single forward pass of the pruned model on the target task's data. Wordnet wup similarity only requires the semantic labels of classes in each task and it is based on the depths of the their corresponding synsets in the Wordnet (Miller, 1995) taxonomies. It does not depend on the pruned model so it is more efficient to compute.

Overlap Between Tasks Let Ω_ℓ^i and Ω_ℓ^j denote the sets of filters/nodes/heads remained in layer-ℓ after running IFP or AHNP for task i and j (when using the same pre-trained model), we measure the overlap of the two sets by intersection over union (IoU) ratio (Jaccard, 1901), i.e., $\text{IoU} = |\Omega_\ell^i \cap \Omega_\ell^j| / |\Omega_\ell^i \cup \Omega_\ell^j|$.

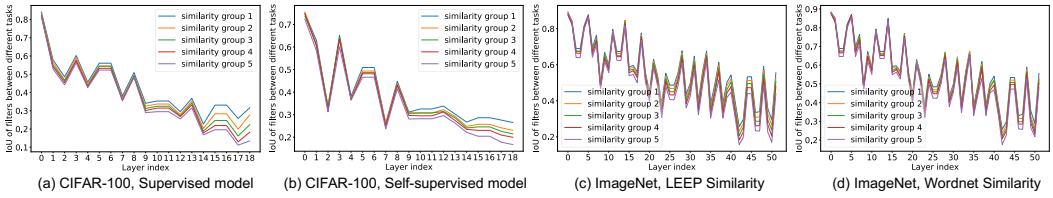
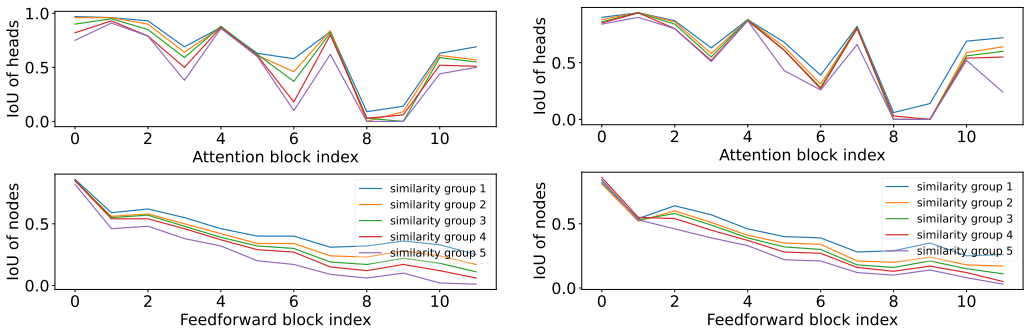


Figure 1: IoU of layers in ResNet between tasks with different similarities.

Fig. 1 (ResNet) and Fig. 2 (ViT) report the IoU of each layer/block for pairs of tasks with different similarities. For each target task, the tasks in the dataset are partitioned into 5 similarity groups according to their LEEP scores or Wordnet similarities to the target task. The similarity decreases from group 1 to group 5.

For all the datasets and architectures, **more similar tasks tend to share more filters/nodes/heads** (larger IoU) between their pruned models. Therefore, for a new task, the pruned models of its nearest tasks preserve many important filters for it and combining them might result in a better and much smaller sub-network to initialize the new task. Moreover, for deeper layers/blocks in both ResNet and ViT, the gap between different similarity groups on the IoU increases because the features are more task-specific in deeper layers. Due to the same reason, for every similarity group, IoU decreases with depth in the overall trend (though fluctuating locally). Furthermore, Fig. 2 shows that the IoU gap between similarity groups defined by the LEEP score is larger than that obtained by



(a) ImageNet, LEEP Similarity (b) ImageNet, Wordnet Similarity

Figure 2: IoU of blocks in ViT between tasks with different similarities.

Wordnet similarity. This indicates that the semantic similarity between class labels might not be as accurate as the LEEP score that takes the pruned model and its learned representations into account.

Algorithm 1 META-VOTE PRUNING (MVP)

Input : Target task i and its training set D_i, pruning ratio r, 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 according to LEEP score or Wordnet similarity;
for ℓ ← 1 to L − 1 do
    Sample (1 − r)n_ℓ filters with probability p(k) (Eq. (1)) and add them to Ω_ℓ;
    for k ∈ Ω_ℓ do
        Initialize filter-k by averaging its parameters of tasks in {j ∈ N^i : k ∈ Ω_ℓ^j};
    end
end
Fine-tune the pruned model for J iterations on D_i.
```

4 META-VOTE PRUNING (MVP)

Inspired by the empirical study above, we propose a simple yet strong baseline "meta-vote pruning (MVP)" (Alg. 1) for non-parametric meta-pruning. The procedure of MVP majority voting is shown in Fig. 3. Given a target task i, MVP draws a sub-network of a pre-trained network by sampling filters/nodes/heads in each layer using majority voting from its nearest tasks N^i and their pruned models. In particular, for each filter-$k \in [n_\ell]$ from layer-ℓ of the pre-trained model, we apply softmax (with temperature τ) to the times of each filter being selected by tasks in N^i, which yields a probability distribution over all the filters $[n_\ell]$, 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-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 which preserve filter-k. MVP then fine-tunes the initialized sub-network for a few iterations on the training set of the target task since MVP targets to keep the computational cost low.

5 EXPERIMENTS

In this section, we conduct extensive experiments on CIFAR-100 (Krizhevsky, 2009) and ImageNet (Ren et al., 2018) over different pre-trained models, which evaluate MVP (Alg. 1) and compare it with SOTA pruning methods under different settings. We study the effect of different meta-pruning iterations, neighbour numbers, and similarity metrics for MVP. All the results show that MVP can

outperform other methods with better performance and higher efficiency. We further validate the strong generalization of MVP by applying it to unseen tasks from Caltech-256.

5.1 Implementation Details

The experiments of MVP are mainly based on the tasks from the dataset introduced in Sec. 3.1. For each setting of experiments, we randomly draw 100 test tasks (i.e., the target task in Alg. 1) from the dataset and treat the rest tasks as training tasks. To evaluate MVP on CNN, we run MVP on the pruned models of ResNet-18 and ResNet-50 for CIFAR-100 and ImageNet respectively. For both these two experiments, we use the meta-pruning iterations of 100, batch size of 128, learning rate of 0.01 and optimizer of SGD with the cosine-annealing learning rate schedule. For experiments of ViT, MVP is applied to the pruned models of ViT for ImageNet. The meta-pruning iterations and batch size are also set as 100 and 128 respectively. Following the setting of training ViT in (Touvron et al., 2021a), we apply a small learning rate of 0.0002 and optimizer of AdamW with the cosine-annealing learning rate schedule. The small number of meta-pruning iterations demonstrates the efficiency of MVP. The target pruning ratio of MVP for all tasks is 0.9. All the results of accuracy shown in this section are averaged over the 100 test tasks.

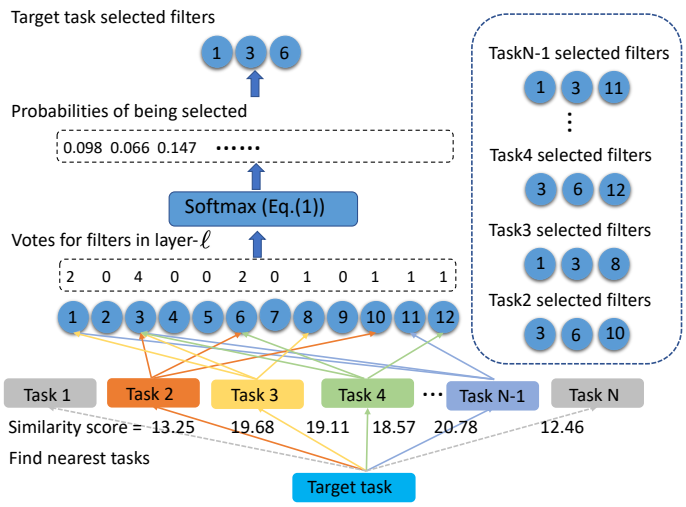


Figure 3: The example of majority voting in MVP. Each similar neighbour task of the target task vote for filters which are reserved by its pruned model. Then, softmax is applied to the votes of all filters in layer-ℓ and filters with more votes have higher probability to be selected by the target task.

5.2 Baseline Methods

We compare MVP with several baselines and SOTA pruning methods. We first implement two baselines to show the advantages of MVP. (1) Conventional pruning. We apply a larger number of pruning iterations to extract pruned models for each target task by IFP or AHNP introduced in Sec. 3.1. This baseline can be regarded to provide the upper bound performance. (2) Random pruning. To validate whether the initialization of MVP makes sense, for each target task, we initialize its sub-network by randomly sampling the same number of nodes/filters/heads as MVP from the pre-trained model. We take this baseline as the one with the lower bound performance.

We also include other SOTA pruning methods as baselines. For MVP on CNN models, we compare MVP with IHT-based Reptile (Tian et al., 2020), a meta-pruning method that uses Reptile (Nichol et al., 2018) and iterative pruning to find better weight-initialization for a pruned meta-model. Given a new task, it fine-tunes the pruned meta-model for a limited number of iterations to obtain the final pruned model. MEST (Yuan et al., 2021) is the SOTA method in sparse training community, which trains a model from a sparse sub-network so that less computation is required. DLTH (Bai et al., 2022) is based on a variant of the Lottery Ticket Hypothesis. It transforms random tickets into winning tickets. We compare MVP with UVC (Yu et al., 2022b) and PoWER (Goyal et al., 2020) on ViT pruning. Unlike AHNP, which prunes heads and nodes, UVC also skips the unimportant layers and blocks in ViT. Unlike parameter pruning, PoWER adopts a dynamic method pruning the input patches of each block for each input sample. For a fair comparison, except for the upper bound baseline, the pruning iterations of all other baselines and MVP are set to 100. And the pruning ratios of all methods are set to 0.9.

Table 1: Comparison between MVP and baseline methods on CNN. The '-SSL' behind each method means applying this method to pruned models extracted from self-supervised pre-trained models. **Bold** and **Bold gray** mark the best and second best accuracy.

Methods	Pruning Iterations	ResNet-18		ResNet-50	
		Acc	FLOPs	Acc	FLOPs
IFP	1000	**87.99±0.47**	14.88(T)	**91.16±0.68**	110.06(T)
IFP-SSL	1000	85.22±0.52	14.88(T)	85.84±0.75	110.06(T)
Random Pruning	100	33.12±6.47	0.43(T)	22.42±3.92	3.16(T)
IHT-based Reptile(Tian et al., 2020)	100	75.23±0.87	0.43(T)	73.40±0.75	3.16(T)
MEST(Yuan et al., 2021)	100	76.28±0.82	0.47(T)	66.25±2.33	3.48(T)
DLTH(Bai et al., 2022)	100	74.46±1.24	4.28(T)	69.33±1.56	31.64(T)
MVP(**ours**)	100	**88.98±0.38**	0.43(T)	**91.80±0.26**	3.16(T)
MVP-SSL(**ours**)	100	86.82±0.13	0.43(T)	85.92±0.26	3.16(T)

5.3 Main Results

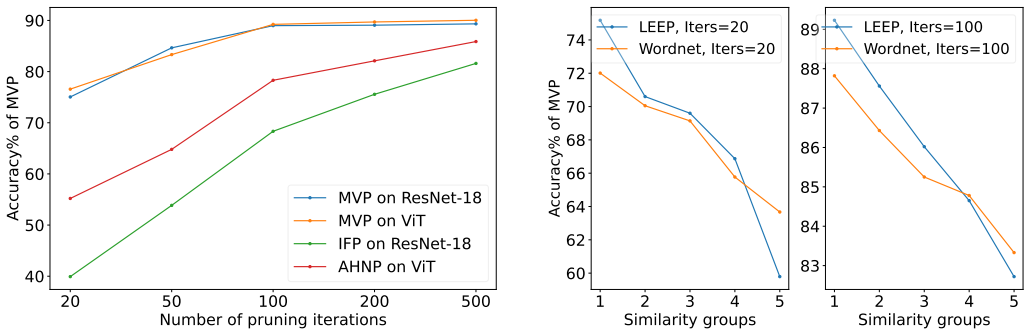
The results of applying MVP to tasks from CIFAR-100(ImageNet) on ResNet-18(ResNet-50) supervised and self-supervised pre-trained model, and the baseline methods are reported in Tab. 1. On both datasets and pre-trained models, MVP outperforms IFP which spends $10\times$ iterations of MVP. Hence, MVP can produce a higher-quality pruned model when using fewer iterations. The results demonstrate that MVP can work well on tasks from both supervised and self-supervised pre-trained models. The random pruning performs much poorer than MVP, which indicates the importance of majority voting from nearest tasks in selecting filters.

We also compare MVP with SOTA pruning methods for CNN. IHT-based Reptile Tian et al. (2020) trains a universal sparse sub-network for all target tasks by applying meta-learning on training tasks. MVP achieves higher accuracy than IHT-based Reptile under the same training iterations, implying that MVP can find an accurate sub-network for each target task as its initialization and improve its performance. MEST Yuan et al. (2021) can speed up pruning by starting training from a well-designed sub-network. As a variant of Lottery Ticket Hypothesis, DLTH Bai et al. (2022) proposes a method to transform any random ticket into the winning ticket. MVP outperforms MEST and DLTH by a large margin because MVP is trained on a sub-network selected using meta knowledge from similar tasks. In contrast, the initial sub-network for MEST or the winning ticket of DLTH does not leverage any prior knowledge about the target task.

Table 2: Comparison between MVP and baseline methods on ViT. **Bold** and **Bold gray** mark the best and second best accuracy

Methods	Pruning Iterations	ViT	
		Acc	FLOPs
AHNP	1000	**89.48±0.62**	81.50(T)
Random Pruning	100	58.71±4.14	3.25(T)
UVC(Yu et al., 2022b)	100	80.30±0.57	26.73(T)
PoWER(Goyal et al., 2020)	100	77.76±1.18	20.86(T)
MVP(**ours**)	100	**89.23±0.49**	3.25(T)

Tab. 2 shows the comparison between MVP and baseline methods on ViT. Similar to the results on pruning CNN, the performance of MVP on ViT is comparable to AHNP that applies much more pruning iterations. The accuracy of random pruning is still much worse. MVP also outperforms SOTA pruning methods developed for ViT. Hence, on ViT, MVP can efficiently produce a small yet high-quality sub-network for each new task by exploiting the nearest tasks' models. The baselines are slower and require more iterations than MVP because they need to re-train the model to achieve a small loss when some parameters or patches are removed. Both UVC (Yu et al., 2022b) and PoWER (Goyal et al., 2020) cannot recover the accuracy under this strong constraint. In contrast, the majority voting in MVP directly produces a small sub-network from similar tasks' models so only a few iterations suffice to reach a downstream task performance comparable to AHNP with 10x iterations.



(a) Effect of Pruning Iterations (b) Effect of Similarities between Tasks

Figure 4: **(a)** Comparison between MVP and conventional pruning methods with different pruning iterations on different architectures. For both ResNet-18 and ViT, MVP converges much faster in a small number of iterations than conventional pruning methods. **(b)** Comparison between LEEP score and Wordnet similarity for MVP with different pruning iterations. From similarity groups 1 to 5, the similarities between tasks decrease. For both similarity metrics, more similar tasks get better performance. LEEP score has a better ability to measure similarities between tasks than Wordnet similarity.

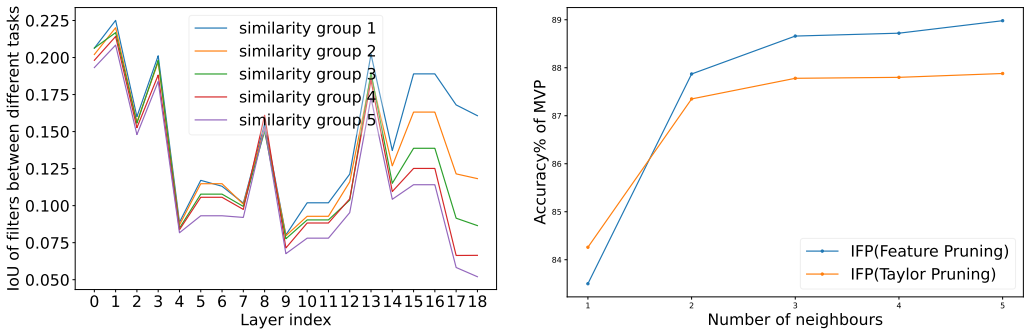
5.4 Ablation Study

Effect of Iteration Numbers Given a new target task and a pre-trained model, MVP can build a well-performed small model in a few iterations, demonstrating its capability in reducing adaptation cost. In the left plot of Fig. 4, we compare MVP with conventional pruning methods using different numbers of iterations. On different architectures of pre-trained models, MVP converges to a high accuracy after nearly 100 iteration. On the contrary, the conventional pruning methods need much more iterations(> 500) to be comparable to MVP. With only ≤ 50 pruning iterations, MVP can reach a reasonable accuracy, while conventional pruning methods perform poorly. These imply that the initialized sub-network obtained by majority voting already contains helpful knowledge from its similar tasks to speed up the training of the pruned model.

Effect of Similarities between Tasks MVP consistently achieves better performance when applied to nearest tasks with the highest similarities. In the right plot of Fig. 4, we compare the LEEP score with the Wordnet similarity and study the effect of applying MVP to neighbour tasks with different similarities. From similarity group 1 to group 5, the similarities between tasks decrease. We find that for both the two similarity metrics, the accuracy of MVP improves significantly when the similarities between tasks increase. When the pruning iterations are small($= 20$), where the initialization of the sub-network is more important, the accuracy of tasks from similarity group 1 leads to similarity group 5 by 15%. Despite the accuracy of similarity group 5 improving when the pruning iterations increase to 100, there is still a gap of 7%. This result indicates that neighbour tasks with high similarities share more knowledge with the target task. In this plot, we also find that tasks in different similarity groups classified by LEEP score show larger differences than Wordnet similarity, implying that LEEP score can better evaluate similarities between tasks. This result is consistent with our observation in the empirical study. The performance of Wordnet similarity is also good and can still be an alternative when the time and computational resources are limited.

Comparison between Pruned Models Extracted by Different Pruning Method In this part, we apply MVP to pruned models extracted by Taylor Pruning (Molchanov et al., 2019) on ResNet-18 for CIFAR-100 tasks, to prove that MVP works well on pruned models extracted by various pruning methods. Taylor Pruning measures the importance of each filter by the effect of removing this filter on final loss. In the left plot of Figure 5, we show the IoU of each layer for pairs of tasks with different task similarities, of which the pruned models are extracted by Taylor Pruning. Consistent with our observation in the empirical study, pruned models with higher similarities share more filters. In the right plot of Figure 5, we investigate the effect of the number of neighbours for MVP. When the number $= 1$, MVP reduces to transfer learning which learns from the pruned model of a single selected similar task. In the plot, when the number of neighbours increase from 1 to 2, the performance improves sharply. This result implies the effectiveness of meta knowledge

from different neighbours. When the number of neighbours ≥ 3, for both Activation Pruning and Taylor Pruning, the accuracy improves little, which indicates that 3 neighbours are enough for MVP to produce a high-quality initialization.



(a) IoU of Layers for IFP(Taylor Pruning)

(b) Effect of Neighbour Numbers

Figure 5: **(a)** IoU of layers in ResNet-18 between tasks whose pruned models are extracted by IFP (Taylor Pruning) and more similar tasks also share more filters, especially in deeper layers. **(b)** Results of applying MVP to pruned models from Activation Pruning and Taylor Pruning over different number of neighbours. MVP(neighbour number ≥ 2) can improve the performance of transfer learning(neighbour number $= 1$) by a large margin when applied to pruned models extracted by different pruning methods.

5.5 Performance on Unseen Dataset

In this section, to validate the generalization of MVP, we apply MVP to produce pruned models for target tasks from Caltech-256 (Griffin et al., 2007) using the pruned models of tasks from ResNet-50 training on ImageNet. The data of Caltech-256 is never seen by the pre-trained model and training tasks in the pruned model dataset. Each target task is defined on 5 classes sampled without replacement from Caltech-256, and each class has about 100 samples. The performance of MVP on Caltech-256 are shown in Tab. 3, which is still comparable to the conventional pruning method using 10x pruning iterations. When the number of pruning iterations of the conventional pruning method decreases, its performance becomes much worse. The results show that MVP can still produce a high-quality initialization for the task from Caltech-256 by majority voting of similar tasks, so that the pruned model can converge quickly with high accuracy. This experiment demonstrates that MVP can be applied to various datasets and generalizes well.

Table 3: Accuracy of applying MVP to unseen tasks from Caltech-256.

Methods	**Pruning Iterations**	**ResNet-50**	
		Acc	FLOPs
IFP	1000	82.40±1.35	110.06(T)
IFP	60	42.90±3.79	6.73(T)
MVP	60	80.72±0.64	1.90(T)

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 an empirical study to investigate the relationship between task similarity and the pruned models of two tasks for different datasets and deep neural networks. The empirical study motivates a simple yet strong baseline for meta-pruning, called "meta-vote pruning (MVP)" (Alg. 1). By extensive experiments on multiple tasks drawn from several datasets under different training settings, we demonstrate the advantages of MVP over other SOTA pruning methods in the region of limited computation and show its potential on reducing carbon footprint of pruning/fine-tuning large networks for billions of edge devices and tasks.

REFERENCES

Yue Bai, Huan Wang, Zhiqiang Tao, Kunpeng Li, and Yun Fu. Dual lottery ticket hypothesis. *arXiv preprint arXiv:2203.04248*, 2022.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

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.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pp. 3690–3699. PMLR, 2020.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

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.

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

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.

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.

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.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

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.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *International Conference on Machine Learning*, pp. 7294–7305. PMLR, 2020.

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

Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *Proceedings of the IEEE/cvf international conference on computer vision*, pp. 377–386, 2021.

Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi, et al. Wordnet:: Similarity-measuring the relatedness of concepts. In *AAAI*, volume 4, pp. 25–29, 2004.

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.

Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12165–12174, 2022.

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

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, volume 139, pp. 10347–10357, July 2021a.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021b.

Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. *arXiv preprint cmp-lg/9406033*, 1994.

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.

Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 2022, 2022a.

Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=9jsZiUgkCZP.

Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34, 2021.

Mingjian Zhu, Yehui Tang, and Kai Han. Vision transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021.

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

A APPENDIX

Algorithm 2 ITERATIVE FILTER PRUNING (IFP)

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

```
for j ← 1 to J do
    if j%h = 0 and |Ω_ℓ| > (1 − r)n_ℓ then
        for ℓ ← 1 to L − 1 do
            Prune p% of filters in Ω_ℓ with the smallest importance score over D_T;
        end
    end
    Apply one SGD step on a mini-batch of D_T to fine-tune the remained filters {θ_{ℓ,i} : ℓ ∈ [L − 1], i ∈ Ω_ℓ} and θ_L;
end
```

A.1 ITERATIVE FILTER PRUNING

The detailed procedure of IFP is described in Algorithm 2. 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 fine-tunes the model for total J 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.

A.2 AUTOMATIC HEAD&NODE PRUNING

The detailed procedure of AHNP is described in Algorithm 3. 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 head/node-i. $S_{\ell,i}$ denote the score for each prunable head/node-i in layer-ℓ. AHNP fine-tunes the model and scores for total J iterations. It prunes the heads/nodes if their scores are smaller than the threshold τ. It stops to prune layer-ℓ if reaching the targeted pruning ratio r. Then, AHNP fine-tunes the pruned model for K iterations.

Algorithm 3 AUTOMATIC HEAD&NODE PRUNING (AHNP)

Input : Pre-trained network $F(\cdot;\theta)$, Task T and training set D_T, Hyperparameters J, K, r, τ
Initialize: $\Omega_\ell \leftarrow [n_\ell]$, the set of heads/nodes preserved in layer-ℓ. $S_{\ell,i} \leftarrow 1$, the score for each prunable head/node in layer-ℓ.

```
for j ← 1 to J do
    for ℓ ← 1 to L − 1 do
        for i ∈ Ω_ℓ do
            Prune the head/node if its score S_{ℓ,i} < τ;
        end
    end
    Stop pruning if reaching the target pruning ratio r.
    Apply one optimization step on a mini-batch of D_T to fine-tune the remained heads/nodes and scores {θ_{ℓ,i}, S_{ℓ,i} : ℓ ∈ [L − 1], i ∈ Ω_ℓ} and θ_L;
end
Remove S, fine-tune the pruned model for K iterations on Di.
```

A.3 RESULTS OF APPLYING MVP TO SUB-TASKS OF DIFFERENT SIZES

In the experiments of applying MVP to 10-classification and 3-classification sub-tasks, the pruning ratio is set to 85% and 95% respectively. The results are shown in Tab.4. From the results we can find that on sub-tasks of different sizes, MVP can always achieve comparable or better performance

Table 4: Results of MVP on sub-tasks of different sizes for CIFAR-100.

Methods	10-**classification**			3-**classification**		
	Pruning Iterations	Acc	FLOPs	Pruning Iterations	Acc	FLOPs
IFP	1500	84.29±0.26	25.48(T)	500	88.75±0.71	5.59(T)
MVP(**ours**)	190	83.53±0.34	1.21(T)	60	89.25±0.23	0.12(T)

than IFP which needs much more computation. MVP is applicable to a variety of tasks of different sizes.

A.4 Comparison of IoU between MVP and random pruning

In Fig.6, besides MVP, we also show the IoU of pruned models extracted by random pruning. When the similarity between tasks is large, the IoU of MVP is much larger than random pruning, implying that these tasks contain lots of relevant information. When the similarity between tasks is small, in the last few blocks, the IoU of nodes is similar to that of random pruning, which indicates that tasks of low similarity share little high-level information with the target task.

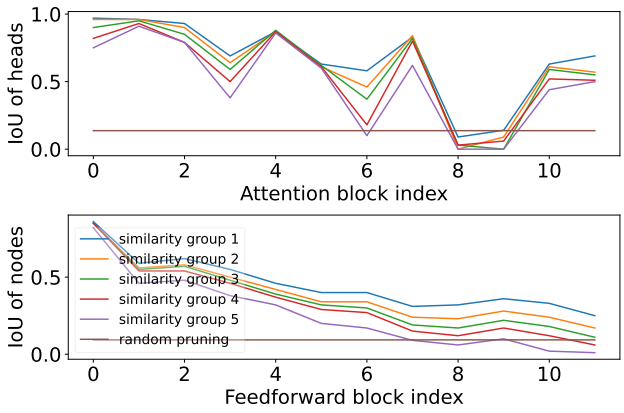


Figure 6: Comparison of IoU between MVP and random pruning.

A.5 The difference of IoU between different similarity groups

In Fig.7, we draw the difference of IoU between tasks of similarity group 1 and similarity group 5 for ResNet-50. As the layer gets deeper, the difference increases.

In Fig.2 of the paper, the average difference of IoU between similarity group 1 and similarity group 5 over all layers is 0.195 and 0.150 respectively for LEEP and Wordnet similarity, which has a large gap. In Fig.4 of the paper, the LEEP score performs a little better than Wordnet similarity in MVP which indicates that models with the larger IoU share more relevant parameters and LEEP has a good ability to find the nearest neighbours.

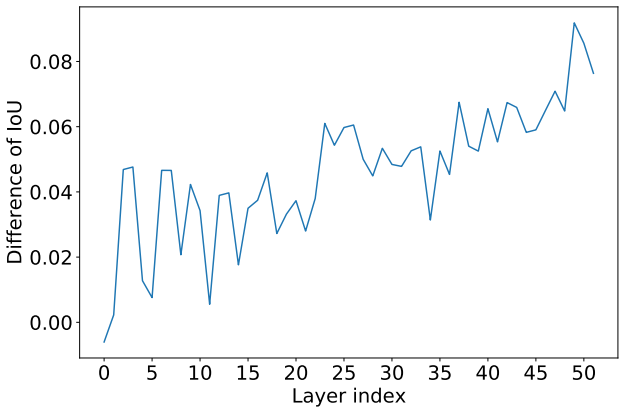


Figure 7: The difference of IoU between tasks of similarity group 1 and similarity group 5 for ResNet-50.