Towards Compute-Optimal Transfer Learning

Massimo Caccia [*] **Alexandre Galashov**[†]

Arthur Douillard[†] **Amal Rannen-Triki**[†] **Dushyant Rao**[†] **Michela Paganini**[†]

Marc'aurelio Ranzato[†] **Razvan Pascanu**[†]

Abstract

The field of transfer learning is undergoing a significant shift with the introduction of large pretrained models which have demonstrated strong adaptability to a variety of downstream tasks. However, the high computational and memory requirements to finetune or use these models can be a hindrance to their widespread use. In this study, we present a solution to this issue by proposing a simple yet effective way to trade computational efficiency for *asymptotic* performance which we define as the performance a learning algorithm achieves as compute tends to infinity. Specifically, we argue that zero-shot structured pruning of pretrained models allows them to increase compute efficiency with minimal reduction in performance. We evaluate our method on the Nevis'22 continual learning benchmark that offers a diverse set of transfer scenarios. Our results show that pruning convolutional filters of pretrained models can lead to more than 20% performance improvement in low computational regimes.

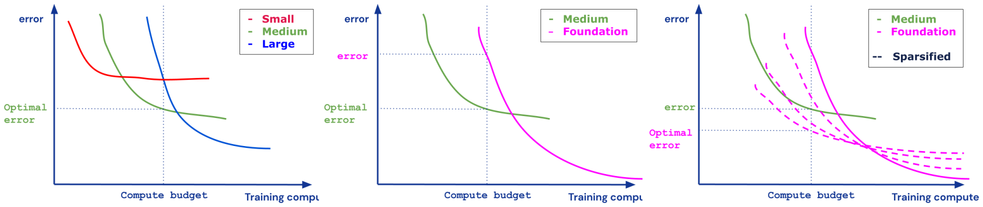
1 Asymptotic Performance versus Computational Efficiency

Sparsity literature (Frankle & Carbin, 2018; Hoefler et al., 2021) has been focused on identifying which weights can be removed from a neural network with minimal impact on its behaviour. A typical approach is to focus on weight magnitude as a proxy, where the weights with smallest magnitude are being pruned. The justification for this comes from taking a Taylor expansion of the pruned model around the dense one. This Taylor expansion depends on the difference between the pruned weights and the dense ones, Δ. The smaller the norm of Δ, the closer is the pruned system to the dense one. Hence pruning the smallest magnitude weights is a cheap and obvious choice.

In this work we want to exploit this intuition by looking at how pruning small magnitude weights affects finetuning given a fixed computational budget. Our hypothesis is that, due to the chain rule, early on in the finetuning process, the learning signal is focused on the large magnitude weights. If we are to compare the finetuned weights with their starting point, most of the change happens on large magnitude weights, while low magnitude weights stay small. We can therefore prune the pretrained model, without affecting the finetuning process, and save compute. However, the larger the computational budget is, and the more gradient steps we do in finetuning, the less likely is that above assumptions still hold. Additionally, there might be a trade-off between model size and computational budget. We therefore show that depending on model size, and computational budget, one can leverage sparsity to gain performance more efficiently than relying on smaller models. Figure 1a exemplifies this intuition.

[*] Current affiliation: Mila - Quebec AI Institute. Work done while interning at DeepMind. email: massimo.p.caccia@gmail.com

[†] DeepMind, London



(a) (b) (c)

Figure 1: Discover the ideal model architecture and size that best fits your target data and training compute budget. The efficiency of different model architectures/sizes varies in terms of asymptotic performance vs compute efficiency. The **left plot** shows the goal of finding a model that optimally fits your compute budget, but a pretrained foundation model is a recommended approach. The **middle plot** highlights that the computational complexity of the foundation model may exceed your budget. The **right plot** presents our solution: zero-shot structured pruning, which increases computational efficiency at the cost of reduced asymptotic performance, allowing the foundation model to fit your compute budget.

An additional important aspect is that unstructured sparsity while in theory should save compute, it rarely can be exploited currently, on traditional hardware and due to the lack of low-level efficient sparse operations. To alleviate this, we focus on structured sparsity that can be readily lead to reduction in computation. We show that depending on the computational budget, there exists a sparsity level that will lead to a better performance/compute tradeoff then finetuning the original dense model.

Structured pruning is a popular technique to reduce the size of deep neural networks and improve computational efficiency. One of the most commonly used structured pruning approaches is channel pruning (He et al., 2017), which reduces the number of channels in a layer. The computational gains from channel pruning are substantial, as the FLOPs decrease at the square of the pruning rate. However, as the representations become smaller, the model has less capacity for learning.

An alternative approach to channel pruning is convolution filter pruning (Li et al., 2016), which prunes at a finer granularity. With convolution filter pruning, the representation size is preserved, but more aggressive pruning is required to achieve the same computational speedup. This is because there are fewer activations to connect between layers in channel pruning. Therefore, for a fixed sparsity value, convolutional pruning enjoys more representations but channel more weights.

Experiments will help us understand the relative merits of channel pruning and convolution filter pruning. For a visual comparison, see Appendix A.

2 Related Work

The closest work to ours is Chen et al. (2021), which examines the existence of sparse subnetworks in *dense* pretrained computer vision models using the lottery ticket hypothesis. With unstructured iterative magnitude pruning (IMP) they find sparse subnetworks that retain the transfer performance of full models. Our work extends this by using structured pruning for computational speedup, zero-shot pruning to save compute, and empirically examining the tradeoff between performance and compute.

In the field of transferring sparse pretrained models, several works have explored the effect of different unstructured pruning techniques during pretraining on transfer performance (Mehta, 2019; Morcos et al., 2019; Paganini & Forde, 2020b; Sabatelli et al., 2020; Sun et al., 2022; Liu et al., 2022). Another line of work focuses on pruning channels (He et al., 2017) and convolution filters (Li et al., 2016). Additionally, structured IMP has been studied in Chen et al. (2022); Paganini & Forde (2020a); Rachwan et al. (2022).

Another work that is relevant to our paper is Frantar & Alistarh (2023), which structurally sparsifies a pretrained GPT without much loss in performance, We refer to Hoefler et al. (2021) for a thourought investigation of sparse neural networks.

3 Experiments

3.1 Experimental Details

In order to study methods in both compute and performance space, we conduct experiments using the new Nevis'22 benchmark (Bornschein et al., 2022). It consists of a stream of over 100 visual classification tasks, sorted chronologically and extracted from papers sampled from computer vision proceedings spanning the last three decades. The stream reflects the research community's understanding of meaningful tasks at any given point in time, and while limited to classification, it includes a diverse range of tasks including OCR, texture analysis, crowd counting, scene recognition, and more. The diversity of tasks is also reflected in the range of dataset sizes, which span over four orders of magnitude.

For our experiments, we focus on the SHORT stream version of Nevis, which consists of 25 tasks, including 16 object recognition tasks and 8 non-object tasks (OCR, scenes, faces, texture, medical). To evaluate approaches that do not start from scratch, Nevis'22 prescribes ImageNet pretraining. Accordingly, we have removed ImageNet from the SHORT stream. The non-object tasks are further considered out-of-distribution (OoD) with respect to ImageNet pretraining.

The backbone of our model is a ResNet50 architecture, taken from the model zoo, a platform of pretrained deep learning models. The ResNet is composed of four ResNet blocks which we label 0 to 3. For the majority of our experiments, we perform pruning locally, i.e., we prune the smaller weights of individual layers. We prune such that the resulting model achieves a desired fraction of the initial required FLOPs required for the forward and backward passes of the model. For weight and convolutional pruning, a sparsity of X% means that X% of the weights or convolution filters have been removed, resulting in only 1-X% of the initial FLOPs. For channel pruning, we instead remove $1 - \sqrt{1 - X\%}$ because the FLOPs decrease at the square of the pruning rate

3.2 Zero-shot structured pruning of dense pretrained models efficiently trades-off performance for compute

In our experiments, we first evaluate the performance of structured pruning in the transfer learning scenario. The results, as shown in Figures 2a and 2b, indicate that convolutional filter pruning performs similarly to unstructured weight pruning, where the latter does not lead to computational efficiency gains on accelerators. Furthermore, compared to channel pruning, convolutional filter pruning is the most efficient way to prune the dense pretrained model. This is especially true for transfer learning tasks that are closer to the pretraining distribution, i.e. objects. It is possible that for target distributions that are closer to the pretraining distribution, more of the learned features are transferable, and therefore, it is necessary to keep all of them.

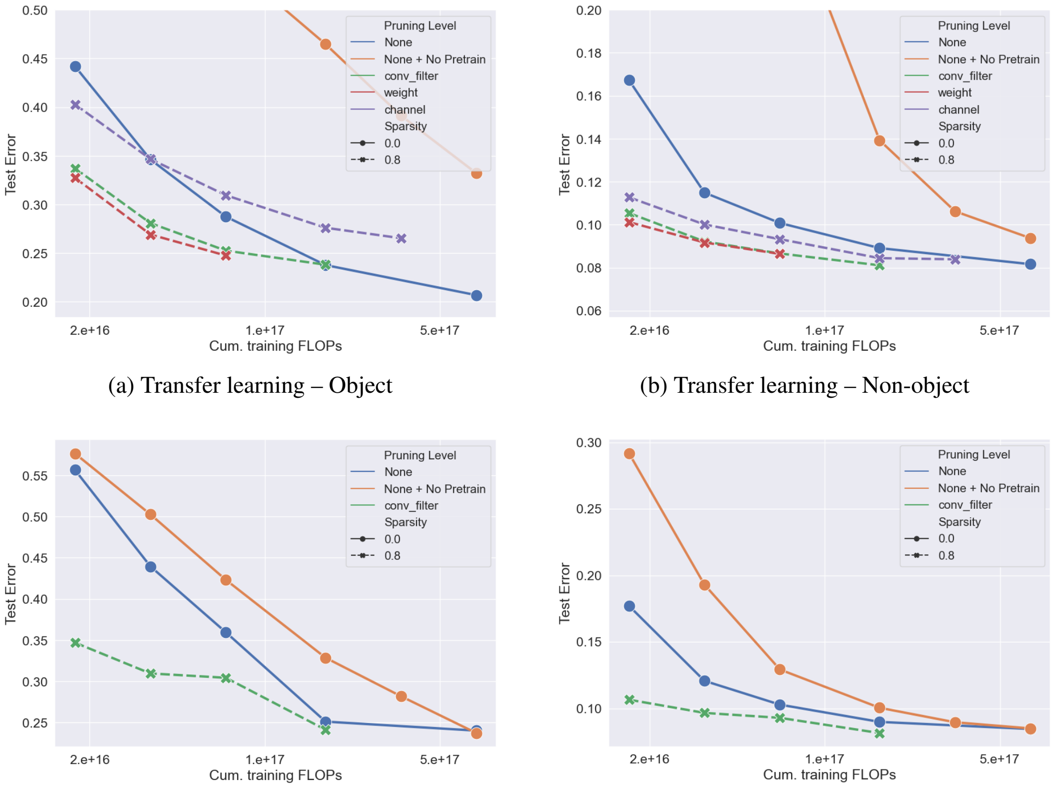
Based on these results, we choose to stick with convolutional pruning for the remaining experiments. We also evaluate the performance of convolutional pruning in the continual learning scenario (see Figure 2c and 2b) and find that it provides an efficient way to trade off performance for compute.

3.3 Zero-shot structured pruning is better than other approaches

There are different approaches to reducing the computational complexity of a pretrained model in transfer learning. The first set of baselines involves finetuning only the last layers of the pretrained model. This approach saves computation by only performing backpropagation on the last layers, which is twice as expensive as the forward pass. The notation `ft_block_Xto3` means finetuning the ResNet blocks X to 3 (the last one), and visual support can be found in the Appendix B.

The results are presented in Figure 3. The proposed method outperforms the compared baselines in transfer learning tasks across different datasets, especially in the object recognition domain. This is because the baselines cannot modify the initial feature representation, which may require more adaptation as the target data distribution deviates from the pre-training distribution.

However, the pruning approach does not achieve superior performance over the baselines in the continual learning scenario on object recognition datasets. This outcome is expected, as the ImageNet pre-training already provides the model with informative representations, leaving limited scope for



(a) Transfer learning – Object

(b) Transfer learning – Non-object

(c) Continual learning – Object

(d) Continual learning – Non-bject

Figure 2: **Zero-shot structure pruning of dense pretrained models efficiently trades-off performance for compute.** In the first set of experiments, we compare different pruning levels on the same pretrained model in transfer learning (top plots). Unstructured Weight pruning achieves the desired tradeoff, but doesn't offer any computational gains on accelerators. Channel prunning is not so efficient in the tradeoff. Alluringly, convolutional pruning achieves almost the same tradeoff as weight pruning. In the second set of experiments (bottom plots), we compare compare convolutional pruning with finetuning the full model in the continual learning setting. Convolutional pruning is again an effective solution.

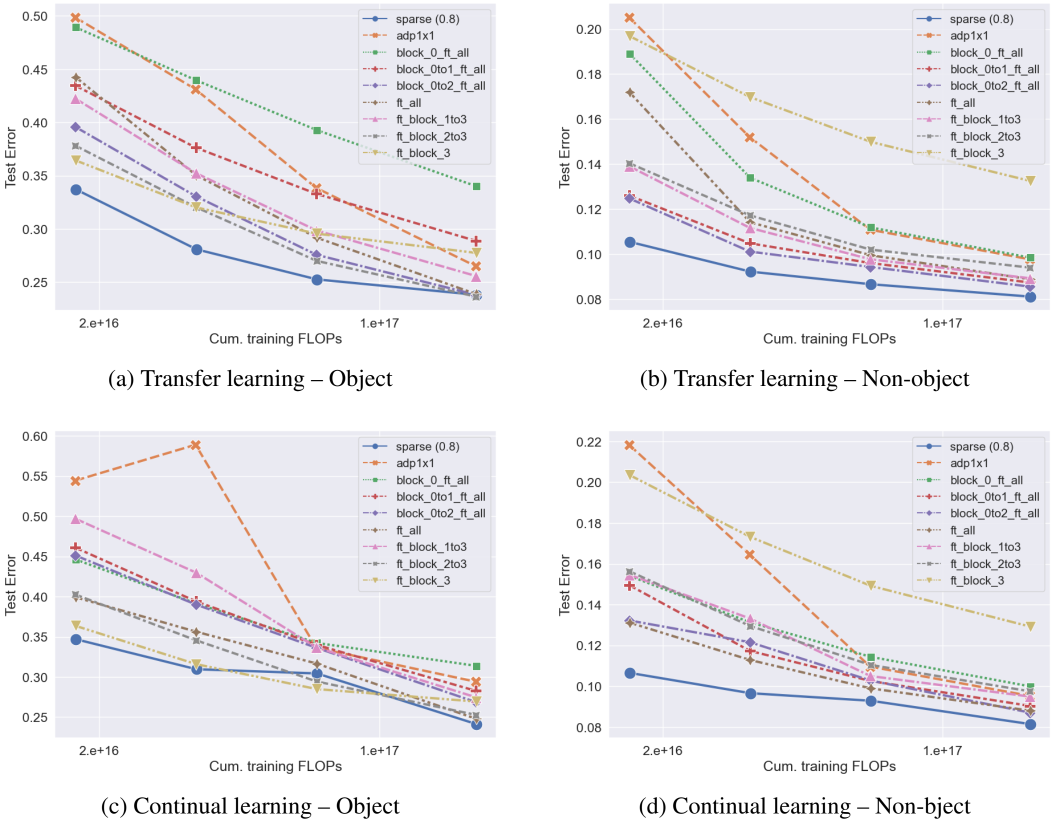
knowledge transfer between tasks. In this case, it may be more effective to freeze the early layers to avoid catastrophic forgetting.

The second set of baselines involves selecting only the first layers and training everything. This approach can modify the earlier representations of the data but does not have the same depth as other approaches, and thus incurs a loss of expressivity. The notation `block_0toX_ft_all` means picking the first X blocks and finetuning everything, with visual support available in the Appendix B.

The results are again displayed in Figure 3. The performance of the `block_0toX_ft_all` baselines is inferior to that of the `ft_block_Xto3`. The proposed zero-shot structured pruning approach demonstrates favorable results across all scenarios. Although the baselines can modify the initial feature representations, the loss in expressivity may render them less effective.

The final baseline utilizes 1x1 convolutional adapters (Rebuffi et al., 2017). At first glance, the use of adapters may appear as a suboptimal trade-off between performance and compute, as they do not reduce the computational demands of the forward and backward passes. However, some literature suggests that this approach may not be computationally efficient . Our experiments confirm that adapters (denoted as `adp1x1`) are not an effective solution for the studied use case, which is unsurprising.

Ablations We have conducted further empirical analysis, namely that of the relationship between pruning rate and the trade-off between performance and compute (refer to Appendix C). Additionally,



(a) Transfer learning – Object

(b) Transfer learning – Non-object

(c) Continual learning – Object

(d) Continual learning – Non-bject

Figure 3: Our results demonstrate that **the zero-shot structured pruning approach is more effective in balancing the trade-off between performance and compute compared to other methods.** In both object and non-object recognition tasks and for transfer and continual learning scenarios, the zero-shot structured pruning approach exhibits superior efficiency. The pruning approach benefits from both full modification of the feature representations and the expressivity of the pretrained model, whereas the baselines are limited to one or the other.

we found that the choice of pruning signal is crucial to the performance of the method, with pruning based on the magnitude of the weights being more effective than random pruning (see Appendix D).

4 Future Work & Discussion

The study of scaling laws, which typically explores the relationship between computation, data, and model size, is a topic of ongoing research. The development of scaling laws for sparsity and transfer would have the potential to predict the ideal level of sparsity for a specific transfer task and computational budget.

Another avenue of research is adaptive pruning, where the pruning process is dynamically adjusted during training via the target loss. Further investigation is also needed to better understand the effects of pretraining on sparsity, including the number of pretraining steps, supervised versus self-supervised training, and model size.

Comparing the results of structured pruning with other methods such as distillation is crucial in evaluating its effectiveness and limitations. Another promising area for future investigation is the use of zero-shot structured pruning in the context of pretrained transformers for natural language processing. In this case, convolutional and channel pruning are analogous to pruning the heads and activations in the transformer layer, respectively.

References

Jorg Bornschein, Alexandre Galashov, Ross Hemsley, Amal Rannen-Triki, Yutian Chen, Arslan Chaudhry, Xu Owen He, Arthur Douillard, Massimo Caccia, Qixuang Feng, et al. Nevis'22: A stream of 100 tasks sampled from 30 years of computer vision research. *arXiv preprint arXiv:2211.11747*, 2022.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16306–16316, 2021.

Tianlong Chen, Xuxi Chen, Xiaolong Ma, Yanzhi Wang, and Zhangyang Wang. Coarsening the granularity: Towards structurally sparse lottery tickets. In *International Conference on Machine Learning*, pp. 3025–3039. PMLR, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Elias Frantar and Dan Alistarh. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1389–1397, 2017.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1):10882–11005, 2021.

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

Yuanxin Liu, Fandong Meng, Zheng Lin, Peng Fu, Yanan Cao, Weiping Wang, and Jie Zhou. Learning to win lottery tickets in BERT transfer via task-agnostic mask training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5840–5857, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.428. URL `https://aclanthology.org/2022.naacl-main.428`.

Rahul Mehta. Sparse transfer learning via winning lottery tickets. *arXiv preprint arXiv:1905.07785*, 2019.

Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *Advances in neural information processing systems*, 32, 2019.

Michela Paganini and Jessica Forde. On iterative neural network pruning, reinitialization, and the similarity of masks. *arXiv preprint arXiv:2001.05050*, 2020a.

Michela Paganini and Jessica Zosa Forde. Bespoke vs. pr\^ et-\a-porter lottery tickets: Exploiting mask similarity for trainable sub-network finding. *arXiv preprint arXiv:2007.04091*, 2020b.

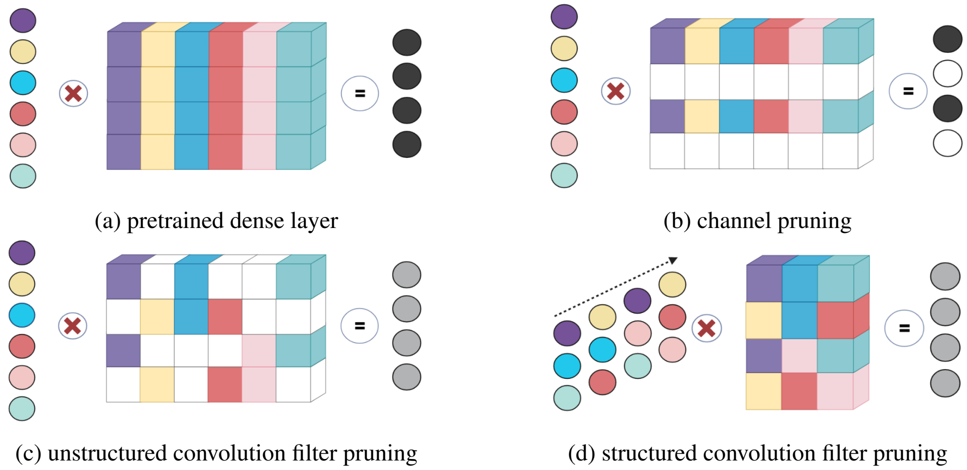
John Rachwan, Daniel Zügner, Bertrand Charpentier, Simon Geisler, Morgane Ayle, and Stephan Günnemann. Winning the lottery ahead of time: Efficient early network pruning. In *International Conference on Machine Learning*, pp. 18293–18309. PMLR, 2022.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

Matthia Sabatelli, Mike Kestemont, and Pierre Geurts. On the transferability of winning tickets in non-natural image datasets. *arXiv preprint arXiv:2005.05232*, 2020.

Xing Sun, Ali Hassani, Zhangyang Wang, Gao Huang, and Humphrey Shi. Disparse: Disentangled sparsification for multitask model compression. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12372–12382, 2022.

A Visual support for pruning type



(a) pretrained dense layer

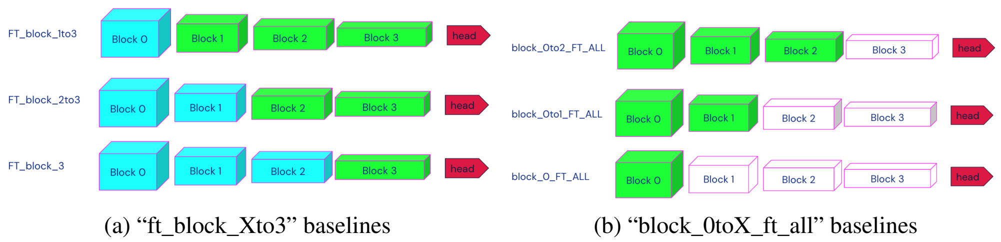
(b) channel pruning

(c) unstructured convolution filter pruning

(d) structured convolution filter pruning

Figure 4: **Visualizing the Impact of Structured Pruning on Convolution Layers.** In this illustration, circles represent input/output feature maps ($H \times W$), cubes symbolize convolution kernels ($K \times K$), and the $\otimes$ operator denotes the convolution. The rows and columns of the convolution layer index the output and input channels, respectively. The pruning rate is set at 50%. The channel pruning approach (b) removes groups of filters responsible for a specific channel, reducing the output representation size and FLOPs by half. In contrast, unstructured convolution filter pruning (c) removes a fraction of filters for each channel while maintaining the output representation size and FLOPs. Structured convolution filter pruning (d) involves restructuring the layer, resulting in a halving of FLOPs, but necessitating the broadcasting of inputs, increasing memory usage. However, the increased memory usage is not prohibitive, as it only occurs locally at the layer level and does not accumulate from one layer to the next.

B Visual support for parameter efficient methods



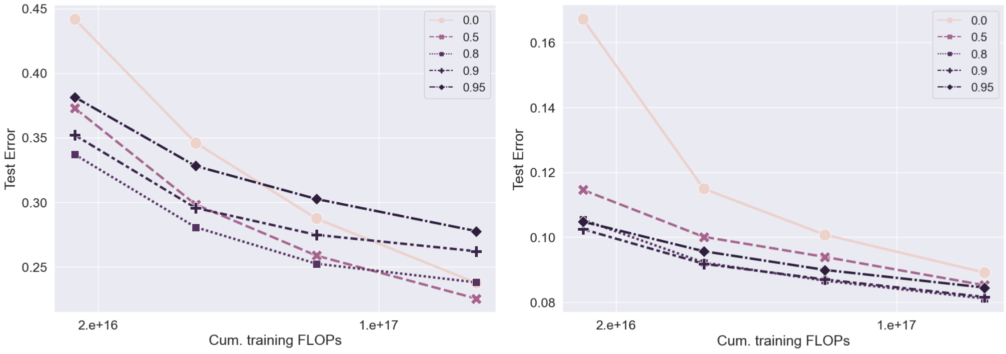
(a) “ft_block_Xto3” baselines

(b) “block_0toX_ft_all” baselines

Figure 5: **Visual support for parameter efficient methods.** The pretrained ResNet is formed of four blocks. Green blocks are trainable, blue are frozen and white are removed.

C Pruning rate controls the asymptotic performance vs compute efficiency trade-off

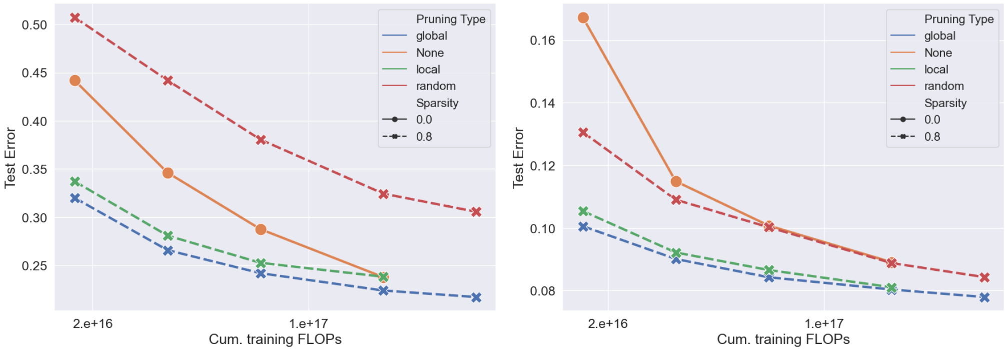
D Pruning signal matters



(a) Transfer Learning - Object

(b) Transfer Learning - Non-object

Figure 6: **Pruning rate modulates the asymptotic performance vs computational efficiency trade-off**



(a) Transfer Learning - Object

(b) Transfer Learning - Non-object

Figure 7: **Pruning signal matters** Randomly pruning the network is not as effective as pruning based on weight magnitude.