How Well Do Sparse ImageNet Models Transfer?

Eugenia Iofinova* IST Austria
Alexandra Peste* IST Austria
Mark Kurtz Neural Magic
Dan Alistarh IST Austria & Neural Magic

Abstract

Transfer learning is a classic paradigm by which models pretrained on large "upstream" datasets are adapted to yield good results on "downstream" specialized datasets. Generally, more accurate models on the "upstream" dataset tend to provide better transfer accuracy "downstream". In this work, we perform an in-depth investigation of this phenomenon in the context of convolutional neural networks (CNNs) trained on the ImageNet dataset, which have been pruned—that is, compressed by sparsifying their connections. We consider transfer using unstructured pruned models obtained by applying several state-of-the-art pruning methods, including magnitude-based, second-order, re-growth, lottery-ticket, and regularization approaches, in the context of twelve standard transfer tasks. In a nutshell, our study shows that sparse models can match or even outperform the transfer performance of dense models, even at high sparsities, and, while doing so, can lead to significant inference and even training speedups. At the same time, we observe and analyze significant differences in the behaviour of different pruning methods. The code is available at: https://github.com/IST-DASLab/sparse-imagenet-transfer.

1. Introduction

The large computational costs of deep learning have led to significant academic and industrial interest in *model compression*, defined roughly as obtaining smaller-footprint models matching the accuracy of larger models. Model compression is a rapidly-developing area, and several general approaches have been investigated, of which pruning and quantization are among the most popular [18, 28].

We focus our present study on *weight pruning*, whose objective is to remove, by setting to zero, as many weights as possible without losing model accuracy. Weight pruning is, arguably, the compression method with the richest history [42] and is currently a very active research topic [28]. Thanks to this trend, a set of fairly consistent accuracy benchmarks has emerged for pruning, along with increasingly efficient computational support [11, 20, 40, 52].

One major goal of model compression is to enable deployment on edge devices. Such devices may naturally encounter different data distributions, so it is tempting to ask how compressed models would perform for *transfer learning*, broadly defined as leveraging information from some baseline "upstream" ("pretrained") task in order to perform better on a "downstream" ("finetuning") task. Specifically, we mainly focus on a prototypical transfer learning setup [36]: starting from models trained and compressed on the ImageNet-1K dataset [60], we refine the resulting models onto several different target tasks. In this context, we examine the question of how well the resulting sparse models transfer. Our motivation is both practical—sparse transfer can provide speedups for both inference and training on the downstream model—and analytical, as we aim to shed light on the impact of sparsity on the resulting features.

Our study will consider two common transfer learning variants: *full finetuning*, where all unpruned weights can be optimized during transfer, and *linear finetuning*, where only the final linear layer of the model is finetuned downstream. While both are popular, we will see that they can lead to different results. We additionally explore *inference-time* speedups using a sparsity-aware inference engine [9], and for the first time examine *training-time speed-up* achievable for linear finetuning via sparse models. Furthermore, we analyze the impact of different pruning methods and task characteristics on transfer performance.

We consider the top-performing pruning methods in terms of ImageNet accuracy, roughly split into three categories. The first is given by *progressive sparsification* methods, which start from an accurate *dense* baseline and proceed to gradually remove weights, followed by finetuning. The prototypical example is *gradual magnitude pruning (GMP)* [17, 22, 23, 69], which uses absolute weight magnitude as the pruning criterion. In addition, we examine WoodFisher pruning [63], which leverages second-order information for highly-accurate pruning.

The second rough category is given by *sparse regu-*

*These authors contributed equally. Correspondence to: {eugenia.iofinova, alexandra.peste}@ist.ac.at

larized training methods, which perform network compression, and possibly network re-growth, during the training process itself. The top-performing methods we consider here are Soft Threshold Reparametrization (STR) [41], Alternating Compressed/DeCompressed Training (AC/DC) [57] and "The Rigged Lottery" (RigL) [12].

The final category comprises Lottery Ticket Hypothesis (LTH)-style methods [5, 6, 14, 15]. These methods emphasize the *discovery* of sparse sub-networks, which can yield good accuracy when re-trained from scratch. Specifically, we consider LTH for transfer (LTH-T) [5], which provides state-of-the-art results among such methods.

We measure the transfer accuracy of sparse ImageNet models obtained via these pruning methods. Our main target application is given by twelve classic transfer datasets, described in Table 2, ranging from general datasets, to more specialized ones. We mainly focus on the classic ResNet50 [26] model, but we extend our analysis to ResNet18, ResNet34 and MobileNet-V1 [31], and we also examine transfer performance for object detection tasks.

Contribution. We present the first systematic study of how different pruning and transfer approaches impact transfer performance. *Our main finding is that sparse models can consistently match the accuracy of the corresponding dense models on transfer tasks.* However, this behaviour is impacted by the following factors: *pruning method* (e.g. regularization vs. progressive pruning), *transfer approach* (full vs. linear), *model sparsity* (e.g. moderate 80% vs. high 98% sparsity), and *task type* (e.g. degree of specialization).

We briefly outline our main conclusions, summarized in Figure 1 and Table 1. For *linear finetuning*, sparse models usually match and can slightly outperform dense models. Yet, this is not true for all pruning methods: *regularization-based* methods perform particularly well, even at high sparsities (e.g. 95%). For *full finetuning*, which generally provides higher accuracies [36], sparse models are also competitive with dense ones, but transfer accuracy is more tightly correlated with accuracy on the ImageNet pretraining task: consequently, less sparse models (e.g. 80%-90% sparsity) tend to be more accurate than sparser ones. Moreover, in this setting we find that *progressive sparsification* methods consistently produce models with higher transfer accuracy, relative to regularization methods. We provide a first analysis of this effect, linking it to structural properties of the pruned models. In addition, we observe the markedly lower accuracy of lottery-ticket approaches, especially at the higher levels of sparsity, e.g. $\geq 90\%$, required for computational speedups.

Given the difference in behaviour between linear and full finetuning, we find that there is currently no single "best" pruning method for transfer. However, using existing methods, one can consistently achieve order-of-magnitude ($\sim$ 90%) compression without loss of accuracy. In turn, these compression levels can lead to speedups of more than $3\times$ on sparsity-enabled runtimes. This suggests that sparse transfer may have significant practical potential.

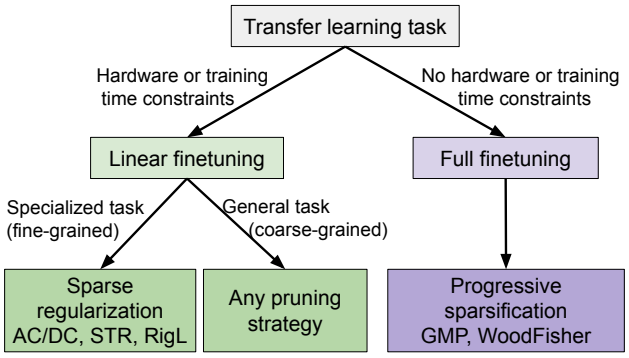

Figure 1. Overview of a suggested decision process when selecting the finetuning and pruning methods to maximize performance and accuracy when doing transfer learning on pruned models.

2. Background and Related Work

2.1. Sparsification Techniques

Recently, there has been significant research interest in pruning techniques, and hundreds of different sparsification approaches have been proposed; please see the recent surveys of [17] and [28] for a comprehensive exposition. We roughly categorize existing pruning methods as follows.

Progressive Sparsification Methods start from an accurate *dense* baseline model, and remove weights progressively in several steps, separated by finetuning periods, which are designed to recover accuracy. A classic instance is gradual magnitude pruning (GMP) [17, 22, 23, 69], which progressively removes weights by their absolute magnitude, measured either globally or per layer. *Second-order* pruning methods, e.g. [16, 24, 42, 63, 65] augment this basic metric with second-order information, which can lead to higher accuracy of the resulting pruned models, relative to GMP.

Regularization Methods are usually applied during model training, via sparsity-promoting mechanisms. These mechanisms are very diverse, from surrogates of ℓ_0 and ℓ_1-regularization [41, 67], to variational methods [53], to methods inspired by compressive sensing mechanisms such as Iterative Hard Thresholding (IHT) [32, 33, 45, 57]. We also consider the "The Rigged Lottery" (RigL) method [12], which achieves close to state-of-the-art ImageNet results by allowing for dynamic weight pruning and re-introduction with long finetuning periods, to be a regularization method.

Lottery Ticket Hypothesis (LTH) Methods [14] start from a fully-trained model, and often apply pruning in a single or in multiple incremental steps to obtain a sparse mask over the weights. They then restart training, but restricted to the

Finetuning	Sparsity	Aircraft	Birds	Caltech-101	Caltech-256	Cars	CIFAR-10	CIFAR-100	DTD	Flowers	Food-101	Pets	SUN397
Linear	0%	49.2 ± 0.1	57.7± 0.1	91.9± 0.1	**84.8± 0.1**	53.4± 0.1	91.2± 0.	**74.6± 0.1**	73.5± 0.2	91.6± 0.1	73.2± 0.	**92.6± 0.1**	60.1± 0.
	80%	55.2± 0.2	58.4± 0.	**92.4± 0.2**	**84.6± 0.1**	58.6± 0.1	**91.4± 0.**	**74.7± 0.1**	**74.4± 0.1**	**93.0± 0.**	**73.9± 0.**	**92.5 ± 0.1**	**60.4± 0.**
	90%	**56.6± 0.1**	**58.7 ± 0.**	**92.5± 0.1**	84.5± 0.1	**60.5 ± 0.1**	91.0 ± 0.	74.3± 0.	73.8± 0.1	**93.0 ± 0.1**	**73.8 ± 0.**	92.0 ± 0.1	59.8 ± 0.1
Full	0%	83.6 ± 0.4	72.4 ± 0.3	**93.5 ± 0.1**	**86.1 ± 0.1**	**90.3 ± 0.2**	**97.4 ± 0.**	**85.6 ± 0.2**	**76.2 ± 0.3**	95.0 ± 0.1	**87.3 ± 0.1**	**93.4 ± 0.1**	**64.8 ± 0.**
	80%	**84.8 ± 0.2**	**73.4 ± 0.1**	**93.7 ± 0.1**	85.4 ± 0.2	**90.5 ± 0.2**	97.2 ± 0.1	85.1 ± 0.1	**75.7 ± 0.5**	**96.1 ± 0.1**	**87.4 ± 0.1**	**93.4 ± 0.1**	64.0 ± 0.
	90%	**84.9 ± 0.3**	72.9 ± 0.2	**93.9 ± 0.3**	84.8 ± 0.1	90.0 ± 0.2	97.1 ± 0.	84.4 ± 0.2	**75.5 ± 0.4**	**96.1 ± 0.1**	**87.3 ± 0.2**	92.7 ± 0.3	63.0 ± 0.

Table 1. Best transfer accuracies at 80% and 90% sparsity for linear and full finetuning, relative to dense transfer. For each downstream task, we present the maximum test accuracy across all sparse methods, highlighting the top accuracy. (We highlight multiple methods when confidence intervals overlap. Results are averaged across five and three trials for linear and full finetuning, respectively.) Note that in all but three cases (all full finetuning), there is at least one sparse model that is competitive with or better than the dense baseline.

given mask. Training restarts either from intialization [14], or by "rewinding" to an earlier point during training of the dense model [6, 7, 15]. (The random mask–initialization combination is thought of as the "lottery ticket".) Rewinding appears to be required for stable results on large datasets such as ImageNet [5, 6, 15].

The above categorization is clearly approximate: for instance, LTH methods could be viewed as a special case of progressive sparsification, where a specific finetuning approach is applied. Moreover, it is not uncommon to combine approaches, such as regularization and progressive sparsification [28]. We provide a comparison of the efficacy of these different approaches in the context of transfer learning, by considering multiple methods from each category. To our knowledge, this is the first such detailed study.

Top-1 test accuracy is the standard metric for comparing pruning methods. We also adopt this metric for examining accuracy in the context of transfer, as no such study exists. Yet, we wish to highlight recent work [29, 30, 44] which examines the robustness of pruned models to input perturbations, as well as the impact of pruning on accuracy of specific segments of the data.

2.2. Transfer Learning and Sparsity

Dense Transfer Learning. A large body of literature has established that, in general, deep learning architectures transfer well to smaller "downstream" tasks, and that full finetuning typically achieves higher accuracy than linear finetuning [36, 61]. (A very recent study [39] suggests that this may be inverted on out-of-distribution tasks.) These findings extend to related tasks, such as object detection and segmentation [51]. Kolesnikov et al. [35] have focused on factors determining the success of transfer learning, and on developing reliable fine-tuning recipes. This has been further extended by Djolonga et al. [10], who concluded that increasing the scale of the original model and dataset significantly improves out-of-distribution and transfer performance, despite having marginal impact on the original accuracy. Salman et al. [61] considered whether *adversarially robust* ImageNet classifiers can outperform standard ones for transfer learning, and find that this can indeed be the case. We complement these studies by examining sparse models and pruning methods.

Sparse Transfer Learning. One of the earliest works to consider transfer performance for pruned models was [54], whose goal was to design algorithms which allow the pruning of a (dense) convolutional model when transferring on a target task. (A similar study was performed by [62] for language models.) By contrast, we focus on the different setting where models have already been sparsified on the *upstream* dataset, and observe higher sparsities that the early study of [54].

Recent work on sparse transfer learning has focused specifically on models obtained via the "Lottery Ticket Hypothesis" (LTH) approach [14], which roughly states that there exist sparsity masks and initializations which allow accurate sparse networks to be trained *from scratch*. There are several works investigating the "transferrability" of models obtained via this procedure for different tasks: for instance, [50] shows that lottery tickets obtained on the CIFAR dataset can transfer well on smaller downstream tasks, while [6, 19] investigate the applicability of lottery tickets for pre-trained language models (BERT), and object recognition tasks, respectively. Mallya et al. [49] considered the related but different problem of adapting a fixed network to multiple downstream tasks, by learning task-specific masks.

The recent work of [5] considers the transfer performance of LTH for transfer, proposing LTH-T, and finding that this method ensures good downstream accuracy at moderate sparsities (e.g., up to 80%). We consider a similar setting, but investigate a wider array of pruning methods (including LTH-T) and additional transfer datasets. Specifically, we are the first to compare LTH-T to competitive upstream pruning methods. We observe that, on full finetuning, most pruning methods consistently outperform LTH-T in terms of downstream accuracy across sparsity levels, by large margins at high sparsities.

3. Sparse Transfer on ImageNet

3.1. Experimental Choices

Transfer Learning Variants. We consider both *full finetuning*, where the entire set of features is optimized over the

Dataset	Number of Classes	Train/Test Examples	Accuracy Metric
SUN397 [66]	397	19 850 / 19 850	Top-1
FGVC Aircraft [48]	100	6 667 / 3 333	Mean Per-Class
Birdsnap [1]	500	32 677 / 8 171	Top-1
Caltech-101 [43]	102	3 060 / 6 084	Mean Per-Class
Caltech-256 [21]	257	15 420 / 15 187	Mean Per-Class
Stanford Cars [37]	196	8 144 / 8 041	Top-1
CIFAR-10 [38]	10	50 000 / 10 000	Top-1
CIFAR-100 [38]	100	50 000 / 10 000	Top-1
Describable Textures (DTD) [8]	47	3 760 / 1 880	Top-1
Oxford 102 Flowers [55]	102	2 040 / 6 149	Mean Per-Class
Food-101 [4]	101	75 750 / 25 250	Top-1
Oxford-IIIT Pets [56]	37	3 680 / 3 669	Mean Per-Class

Table 2. Datasets used as downstream tasks for transfer learning.

downstream dataset, and *linear finetuning*, where only the last layer classifier is finetuned, over sparse models. In the former case, with the exception of the final classification layer, only the nonzero weights of the original model are optimized, and the mask is kept fixed, as well as the batch normalization (BN) parameters.

We do not consider from-scratch training and pruning on the downstream task, for two reasons. First, from-scratch training is often less accurate than (dense) transfer learning in the same setting [36, 51]. As our experiments will show, transfer from *sparse models* can often match or even slightly outperform transfer from *dense models*. Second, since training from scratch is typically less accurate than transfer [36], it seems unlikely that training and pruning from scratch will outperform sparse transfer. We give evidence for this claim in Appendix A. One practical advantage of this approach is not needing hyper-parameter tuning with respect to compression on the downstream dataset.

Network Architectures. Our study is based on an in-depth analysis of sparse transfer using the ResNet50 architecture [26]. This architecture has widespread practical adoption, and has been extensively studied in the context of transfer learning [36, 61]. Importantly, its compressibility has also emerged as a consistent benchmark for CNN pruning methods [28]. We further validate some of our findings on ResNet18, ResNet34 and MobileNet [31] architectures. In addition, we investigate transfer between two classical object detection tasks, MS COCO [46] and Pascal VOC [13], using variants of the YOLOv3 architecture [59].

Sparsification Methods. For our study, we chose the pruning methods providing top validation accuracy for each method type in Section 2.1. For *progressive sparsification methods*, we use the leading WoodFisher [63] and Gradual Magnitude Pruning (GMP) [17,22,23,69] methods. For *regularization methods*, we consider the leading Soft Threshold Weight Reparametrization (STR) [41], and Alternating Compression/Decompression (AC/DC) [57] methods. Additionally, we include the "The Rigged Lottery" (RigL) method [12] with Erdős-Rényi-Kernel (ERK) weight density. Compared to STR and AC/DC, RigL extends the training schedules on ImageNet by up to 5x, and does *sparse training* for most of the optimization steps. We consider both the standard version or RigL (RigL ERK 1x), and the variant with 5x training iterations (RigL ERK 5x). Finally, for *LTH Methods*, we consider the LTH-for-Transfer (LTH-T) method of [5], which precisely matches our setting. In this version, the authors apply the masks obtained through progressive sparsification methods directly to the original trained ImageNet dense model, and evaluate the transfer accuracy of this masked model through full finetuning on different downstream tasks.

Sparsity	Method	Original Validation	Reassessed Labels	ImageNetV2 (Average)
0%	Dense	76.8%	83.1%	72%
80%	AC/DC	76.2%	82.9%	71.8%
	STR	75.5%	81.9%	70.3%
	WoodFisher	**76.7%**	**83.2%**	**72.3%**
	GMP	76.4%	82.9%	71.6%
	RigL ERK 1x	74.8%	81.3%	70.2%
	RigL ERK 5x	75.8%	81.6%	70.6%
90%	AC/DC	75.2%	82.2%	70.6%
	STR	74.0%	80.9%	69.1%
	WoodFisher	75.1%	**82.4%**	**71.1%**
	GMP	74.7%	81.6%	70.1%
	RigL ERK 1x	73.2%	80.0%	67.9%
	RigL ERK 5x	**75.7%**	81.9%	70.6%
95%	AC/DC	73.1%	80.4%	68.6%
	STR	70.4%	77.9%	66.0%
	WoodFisher	72.0%	79.8%	67.6%
	RigL ERK 1x	70.1%	77.5%	65.5%
	RigL ERK 5x	**74.0%**	**80.8%**	**69.0%**

Table 3. Accuracy of the pruning methods we use, at different sparsity levels, evaluated on different ImageNet validation sets.

We focus on *unstructured* pruning, as these methods are the most studied in the pruning literature, have well established benchmarks, and achieve the best trade-off between accuracy and compression. We include results for full finetuning from models with structured sparsity in Appendix J, showing that, given a fixed accuracy level upstream, structured-sparse models tend to underperform unstructured-sparse models for transfer.

When available, we use original sparse PyTorch checkpoints, and the exact architectures used by the upstream models. However, since the STR and RigL models were trained using label smoothing, which has been shown in [36] to decrease transfer accuracy, we used retrained versions of these models on ImageNet, without label smoothing. The results we discuss in the following sections are for these versions, which indeed perform better, particularly on linear finetuning (see Appendix I). We manually ported RigL checkpoints from TensorFlow to PyTorch (see Table 3 for all ImageNet results).

Downstream tasks and training. We follow [61] in using the twelve standard transfer benchmark datasets described in Table 2, which span several domains and sizes. We transfer all parameters of the upstream model except for the last

(fully connected) layer, which is adjusted to the number of classes in the downstream task, using Kaiming uniform initialization [25], and kept dense. This may slightly change the sparsity of the model, as in some cases the final layer was sparse. As a convention, when discussing sparsity levels, we refer to the *upstream* checkpoint sparsity. We provide full training hyperparameters in Appendix B.

Performance metrics. The main quantity of interest is the top-1 validation accuracy on each transfer task, measured for all the pruned models, as well as for the dense baselines. In some cases, we use the mean per-class validation accuracy following the convention for that dataset (see Table 2). To determine the overall "transfer potential" for each pruning method, we further present the results aggregated over the downstream tasks. Since datasets we use for transfer learning have varying levels of difficulty, as reflected by the wide range of transfer accuracies, we compute for each downstream task and model the *relative increase in error* over the dense baseline. Specifically, if B is the baseline dense model, then for every downstream task D and sparse model S we define the relative increase in error as $\alpha_{D,S} = \frac{err_{D,S} - err_{D,B}}{err_{D,B}}$, where $err_{D,S}$ is the error corresponding to the top validation accuracy for model S trained on dataset D. For each pruning method and sparsity level, we report the mean and standard error of $\alpha_{D,S}$, computed over all downstream tasks.

We also examine the computational speedup potential of each method, along with its accuracy. For *inference-time* speedups, our findings are in line with previous work, e.g. [11,57,63]. We will therefore focus on the *training-time* speedup potential in the case of *linear finetuning*, which are usually close to inference-time speedups, as the only difference is the training time of the classifier layer.

3.2. Validation Accuracy on ImageNet Variants

To set a baseline, we first examine accuracies on the original ImageNet validation set, and on different versions of this validation set. Namely, we use the ImageNet "reassessed labels" [2], where the original ImageNet validation images are re-assessed by human annotators. We also use three different ImageNetV2 validation sets [58], where the new images with a similar data distribution are gathered based on different criteria. We report the average ImageNetV2 accuracy across these three variants in Table 3.

Discussion. We observe that RigL ERK 5x outperforms all methods on the original validation set at 90% and 95% sparsity, followed by AC/DC, GMP and WoodFisher. At 80% sparsity, WoodFisher has the best original validation accuracy, followed closely by GMP and AC/DC. However, despite the gap in original validation accuracy between RigL ERK 5x and other methods, the results on new variants of the validation set still reveal some interesting patterns. For example, WoodFisher outperforms all methods at 80% and 90% sparsity on the reassessed labels, followed closely by AC/DC. This is true also for ImageNetV2, where WoodFisher outperforms all methods at 80% and 90% sparsity. At 95% sparsity, however, RigL ERK 5x outperforms all methods considered, including on the reassessed labels and ImageNetV2, and is followed by AC/DC. Generally, the accuracies on the reassessed labels and ImageNetV2 correlate well with those on the original images, which suggests that top performing methods can "extrapolate" well.

3.3. Linear Finetuning

Next, we study the transfer performance of different types of pruning methods in the scenario where only the linear classifier "on top" of a fixed representation is trained on the downstream task. Specifically, we study the simple setup where the features prior to the final classification layer of the pre-trained model are extracted for all samples in the transfer dataset and stored into memory for use when training the downstream linear classifier. Although this approach typically results in lower accuracy relative to full finetuning [36, 61], it has significant practical advantages. Specifically, the features can be precomputed, which eliminates the forward passes through the pretrained network. In this setup, we do not apply any data augmentation on the transfer samples and we use the Batch Normalization statistics of the pretrained network on ImageNet.

We optimize the linear classifier using SGD with momentum, weight decay and learning rate annealing, following [61]. (The results are typically well-correlated with those obtained when using data augmentation during training, or using different optimizers [36]). In Section 3.6, we show that training speed-ups can also be obtained in an online learning setup, where new samples are executed through the backbone network, by taking advantage of the backbone sparsity.

The results for linear finetuning are shown in Figure 2, and Appendix Table C.1. We exclude the LTH-T method from this analysis, as it is designed for full finetuning, and its transfer accuracy in the linear scenario is indeed very low (see Appendix Table C.1).

Overall, the results clearly show that the choice of pruning strategy on the upstream task can result in significant differences in performance on downstream tasks. These differences are more apparent for specialized downstream tasks, with fine-grained classes. For example, consider Aircraft, where for 80% sparse models we see a 15% gap in top-1 test accuracy between the best-performing sparse models (AC/DC and RigL, 55%) and the worst-performing one (WoodFisher, 40%).

Following this observation, we study the correlation between the downstream task difficulty and relative increase in error for different pruning strategies. For this purpose, we use the difference in top-1 validation accuracy between

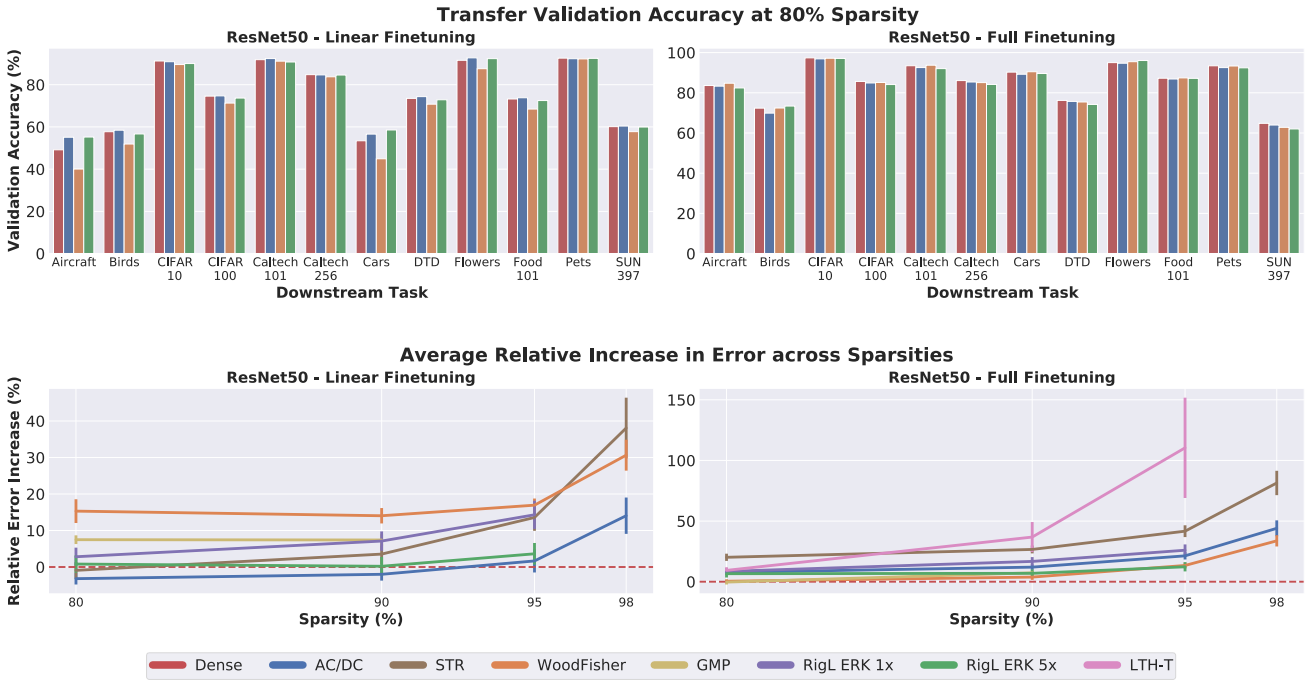

Figure 2. (top row) Validation accuracy for selected pruning strategies at 80% sparsity. (bottom row) Average increase in validation error relative to the dense baseline; lower values are better. Best viewed in color.

full and linear finetuning on the dense backbone as a proxy for the difficulty of a downstream task. Intuitively, a small gap between full and linear finetuning would suggest that the upstream features are directly transferable, and thus the downstream task can be considered "easy". Conversely, a large gap would indicate that the pre-trained features are not enough to capture the internal representation of the data, making the downstream task more "difficult". Additionally, we categorize the downstream tasks into *general* (Caltech-101/256, CIFAR-10/100, DTD, SUN397) vs. *specialized* (Aircraft, Birds, Cars, Flowers, Food-101, Pets); this is similar to previous work [36]. Figure 3 suggests that specialized datasets tend to have higher difficulty scores.

Following this definition and categorization, we measure, for each pruning strategy, the relative error increase over the dense model against the task difficulty. Figure 3 shows the behavior for all pruning methods considered at 80% and 90% sparsity. Interestingly, we observe a trend for *regularization methods* (AC/DC, STR, RigL) to improve over the dense baseline with increased task difficulty, which is more apparent at higher sparsity (90%). In contrast, progressive sparsification methods (GMP, WoodFisher) do not show a similar behavior. This suggests that regularization pruning methods are a better choice for linear transfer (sometimes even surpassing the dense performance) when the downstream task is more specialized or more difficult.

Another particularity of linear finetuning from sparse models is that the sparsity level is not highly correlated with the performance on the downstream tasks. This is apparent, for example, for AC/DC and RigL, where, despite the 1-2% gap in ImageNet accuracy between the 80% and 90% sparse models, the relative error with respect to the dense baseline stays quite flat. A similar trend can be observed for other pruning methods as well. However, extremely sparse models (98%) tend to perform worse, probably due to feature removal and degradation.

In summary, we observe that 1) some sparsification methods can consistently match or even sometimes outperform dense models; 2) there is a correlation between transfer performance for regularization-based methods and downstream task difficulty; and 3) higher sparsity is not necessarily a disadvantage for transfer performance.

3.4. Full Finetuning

We now consider the *full finetuning* scenario. Here, we re-initialize the final classification layer and fix it as dense, then finetune the unpruned weights so that the network is sparse throughout training. The results are summarized in Figure 2, and detailed in Appendix Table C.2.

Similar to linear finetuning, we see substantial performance variations among pruning strategies when transferred to the downstream tasks. Typically, progressive sparsification methods (WoodFisher, GMP) tend to transfer better than regularization and lottery ticket methods. The dif-

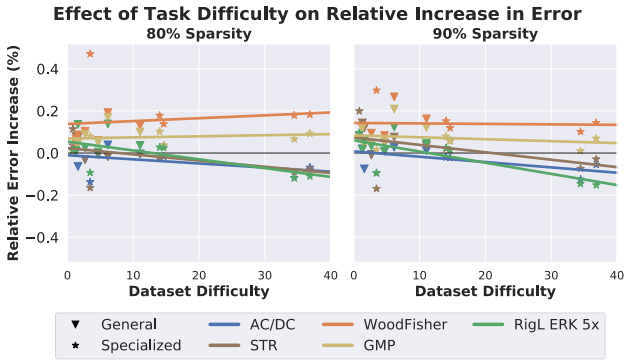

Figure 3. Effect of task difficulty on various pruning strategies for transfer with linear finetuning. Best viewed in color.

ferences in test accuracy, measured at the same sparsity level, are typically small, on the order of 1–3%; the exception is LTH-T, which is competitive at low sparsity (80%) but incurs severe accuracy drops at sparsities $\geq 90\%$.

In contrast to linear finetuning, we see a consistent trend of decreasing quality with increased sparsity. This is not surprising, since full finetuning can take advantage of the additional parameters available in denser models to better fit the downstream data. Nevertheless, *progressive sparsification* methods (GMP and WoodFisher) result in downstream performance nearly on par with dense models at 80% and 90% sparsity. These methods show better performance than regularization-based methods (AC/DC, STR, and RigL), a direct reversal of the results of linear finetuning.

For specific downstream tasks, however, there is considerable variability—while WoodFisher and GMP are consistently the top or near-top performing models across all tasks, other methods show considerable task dependence. For instance, while AC/DC is the top performing method across different sparsities for three of the twelve tasks (SUN397, Caltech-256, and DTD), it shows a considerable gap compared to the best-performing methods on Aircraft, Cars, and CIFAR-10. Generally, STR performs worse on full finetuning, compared to other regularization methods. Finally, RigL ERK 1x performs roughly on par with AC/DC despite having a lower validation accuracy on ImageNet; however, the extended training of RigL ERK 5x gives the tansfer accuracies a considerable boost, putting RigL ERK 5x almost on par with WoodFisher, especially at higher sparsities. This finding opens the intriguing possibility that *extended* training may be beneficial for pruning methods in the full finetuning regime. Finally, LTH-T shows fairly competitive performance at 80% sparsity, but its transfer accuracy declines dramatically on six of the twelve datasets (SUN397, Caltech-101, Caltech-256, DTD, Flowers, and Pets) as sparsity increases. Since the LTH-T model relies mainly on transferring the sparsity mask across tasks, this suggests that the additional information present in the *weights*, leveraged by other methods, may be beneficial.

In sum, if the goal is to perform full finetuning on downstream tasks, then *progressive sparsification* methods are a good choice. They consistently outperform regularization methods across a wide range of tasks, and offer comparable performance to the dense backbone at 80% and 90% sparsity.

3.5. Discussion

The results of the last two sections show an intriguing performance gap between pruning methods, depending on the transfer approach. Investigating further, we examine the sparse structure of the resulting pruned models by measuring the percentage of convolutional filters that are completely pruned away during the training phase of the sparse ResNet50 backbones on the original ImageNet dataset. We observe that AC/DC has a large number of channels that are fully removed during ImageNet training and pruning, on average 2-4 more at 80% and 90% sparsity, compared to other models; this results in fewer features that can be trained during full finetuning. By contrast, the sparsity in GMP and WoodFisher is less structured and thus can express additional features, which can be leveraged during finetuning. We present the exact numbers for all methods in the Appendix E and further illustrate this in Appendix J, where we fully finetune from models with structured sparsity.

In the case of linear finetuning, we hypothesize that the accuracy reversal in favor of AC/DC can be attributed to a regularizing effect, which produces more "robust" features. The same effect appears to be present in RigL ERK 5x at 95% sparsity, which also has significantly many fully-pruned filters.

3.6. Training Speed-Up using Linear Finetuning

One of the main benefits of sparse models is that they can provide *inference speed-ups* when executed on sparsity-aware runtimes [11, 40, 57, 63]. For linear finetuning, this can also imply *training time* speed-ups, since the sparse backbone is fixed, and only used for inference. We illustrate this in an "online learning" setting, where training samples arrive dynamically at the device. We first compute the corresponding features using the sparse backbone. Then, we use these features to train the linear classifier. Thus, the forward pass can benefit from speed-ups due to sparsity.

To measure these effects, we integrated the freely-available sparsity-aware DeepSparse CPU inference engine [9, 40] into our PyTorch pipeline. Specifically, we use sparse inference for online feature extraction. We report overall training speedup, in terms of average training time per epoch on the downstream task, divided by the average training time using the dense baseline. We use batch size

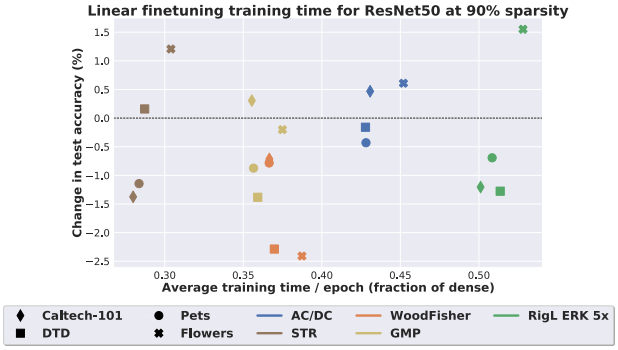

Figure 4. Average epoch time vs. gap in validation accuracy, compared to the dense baseline. Results are shown for four different downstream tasks, using linear finetuning from ResNet50 90% sparse models. Lower is better; best viewed in color.

64 and data augmentation; otherwise, hyperparameters are identical to the linear finetuning experiment in Section 3.3. We execute on an Intel E5-1650 CPU with 12 cores, which would be similar in performance to a recent laptop CPU. The speed-ups we report are proportional to the inference speed-ups of the respective sparse backbone models. The only difference is the cost of optimizing the last layer, which varies in size with the number of classes.

Figure 4 shows results on four downstream tasks, Pets, Flowers, DTD and Caltech-101, where the backbone ResNet50 models have 90% sparsity. We report the training speed-up vs. the difference in validation accuracy, compared to the dense baseline. Results show that *using sparse backbones can reduce training time by 2-3x for linear transfer, without negative impact on validation accuracy*. Additional numbers are provided in Appendix D.4.

4. Extensions

ResNet18/34 and MobileNet Experiments. We have also executed a subset of the experiments for ResNet18, ResNet34 and MobileNetV1 [31] models trained on ImageNet. These results largely validate our analysis above, and are therefore deferred to the Appendix. Specifically, regularization-based methods also match or slightly outperform dense ones on linear transfer. However, for MobileNetV1, we observe that sparse models can match the dense baseline transfer performance only at lower sparsities (up to 75%), probably due to the lower parameter count.

Structured Sparsity Experiments. We also performed full finetuning using models with *structured sparsity*, for both ResNet50 and MobileNet. Our findings, presented in Appendix J, show that structured-sparse models tend to transfer worse compared to unstructured methods.

Sparse Transfer using YOLO. We also examined transfer performance between YOLO V3 [59] and YOLO "V5" [64] models for object detection, trained and pruned on the COCO dataset [46], which are then transferred to the VOC dataset [13] using full finetuning. Table 4 presents results in terms of mean Average Precision (mAP@0.5). Results show a strong correlation between accuracy on the original COCO dataset and that on VOC, confirming our claims. We observed similar trends in a segmentation setup, which we cover in Appendix K.

Architecture Pruning	YOLOv3 90% Sparsity	YOLOv5S 75% Sparsity	YOLOv5L 85% Sparsity
COCO Dense	64.2	55.6	65.4
COCO Pruned	62.4	53.4	64.3
VOC Dense Transfer	86.0	83.73	90.0
VOC Pruned Transfer	84.0	81.72	89.35

Table 4. Accuracies for Sparse Transfer from COCO to VOC.

5. Conclusions and Future Work

We performed an in-depth study of the transfer performance of sparse models, and showed that pruning methods with similar accuracy on ImageNet can have surprisingly disparate Top-1 accuracy when used for transfer learning. In particular, regularization-based methods perform best for linear finetuning; conversely, progressive sparsification methods such as GMP and WoodFisher tend to work best when full finetuning is used. One limitation of our study is that it only investigated accuracy as a measure of performance for transfer learning tasks. Additional research is needed towards designing pruning strategies with good performance across *both* linear and full finetuning, and towards considering metrics past Top-1 accuracy, such as bias and robustness. Another limitation is that we considered a (standard) fixed set of transfer datasets; our study should be extended to other, more complex transfer learning scenarios, such as distributional shift [34]. Further investigation could also systematically examine other types of compression, such as quantization and structured pruning, potentially in conjunction with unstructured pruning, which was the focus of our current study. Other interesting areas for future work would be understanding the performance gap between full finetuning and linear finetuning, and realizing training speedups for sparse full finetuning, by taking advantage of the fixed sparsity in the trained model.

Acknowledgments

The authors would like to sincerely thank Christoph Lampert and Nir Shavit for fruitful discussions during the development of this work, and Eldar Kurtic for experimental support. EI was supported in part by the FWF DK VGSCO, grant agreement number W1260-N35, while AP and DA acknowledge generous support by the ERC, via Starting Grant 805223 ScaleML.

References

[1] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2019–2026, 2014. 4

[2] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with ImageNet? *arXiv preprint arXiv:2006.07159*, 2020. 5

[3] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *ICCV*, 2019. 22

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, 2014. 4

[5] 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. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 4

[6] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained BERT networks. *arXiv preprint arXiv:2007.12223*, 2020. 2, 3

[7] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. *arXiv preprint arXiv:2106.01548*, 2021. 3

[8] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 4

[9] DeepSparse. NeuralMagic DeepSparse Inference Engine, 2021. 1, 7

[10] Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander D'Amour, Dan Moldovan, et al. On robustness and transferability of convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[11] Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. Fast sparse convnets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14629–14638, 2020. 1, 5, 7

[12] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning (ICML)*, 2020. 2, 4

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010. 4, 8

[14] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 3

[15] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019. 2, 3

[16] Elias Frantar, Eldar Kurtic, and Dan Alistarh. M-FAC: Efficient matrix-free approximations of second-order information. *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2, 17, 18

[17] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019. 1, 2, 4

[18] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021. 1

[19] Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry S Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 762–771, 2021. 3

[20] Graphcore. Graphcore Poplar SDK 2.0, 2021. 1

[21] Gregory Griffin, Alexander D. Holub, and Pietro Perona. The Caltech 256. *Caltech Technical Report*, 2006. 4

[22] Masafumi Hagiwara. A simple and effective method for removal of hidden units and weights. *Neurocomputing*, 6(2):207 – 218, 1994. Backpropagation, Part IV. 1, 2, 4

[23] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2015. 1, 2, 4

[24] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299. IEEE, 1993. 2

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015. 5

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2, 4

[27] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, 2018. 22

[28] 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. *arXiv preprint arXiv:2102.00554*, 2021. 1, 2, 3, 4

[29] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019. 3

[30] Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. Characterising bias in compressed models. *arXiv preprint arXiv:2010.03058*, 2020. 3

[31] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 4, 8, 17

[32] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-KAST: Top-K always sparse training. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[33] Xiaojie Jin, Xiaotong Yuan, Jiashi Feng, and Shuicheng Yan. Training skinny deep neural networks with iterative hard thresholding methods. *arXiv preprint arXiv:1607.05423*, 2016. 2

[34] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021. 8

[35] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (BiT): General visual representation learning. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[36] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better ImageNet models transfer better? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2661–2671, 2019. 1, 2, 3, 4, 5, 6, 13, 20

[37] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 4

[38] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4, 12

[39] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022. 3

[40] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Bill Nell, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *International Conference on Machine Learning (ICML)*, 2020. 1, 7

[41] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning (ICML)*, 2020. 2, 4

[42] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Conference on Neural Information Processing Systems (NeurIPS)*, 1990. 1, 2

[43] Fei-Fei Li, R. Fergus, and Pietro Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004. 4

[44] Lucas Liebenwein, Cenk Baykal, Brandon Carter, David Gifford, and Daniela Rus. Lost in Pruning: The Effects of Pruning Neural Networks beyond Test Accuracy. *Conference on Machine Learning and Systems (MLSys)*, 2021. 3

[45] Tao Lin, Sebastian U Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[46] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. 4, 8

[47] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. 12

[48] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 4

[49] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[50] Rahul Mehta. Sparse transfer learning via winning lottery tickets. *arXiv preprint arXiv:1905.07785*, 2019. 3

[51] Thomas Mensink, Jasper Uijlings, Alina Kuznetsova, Michael Gygli, and Vittorio Ferrari. Factors of influence for transfer learning across diverse appearance domains and task types. *arXiv preprint arXiv:2103.13318*, 2021. 3, 4

[52] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021. 1

[53] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning (ICML)*, 2017. 2

[54] 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. 3

[55] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1447–1454, 2006. 4

[56] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 4

[57] Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. AC/DC: Alternating Compressed/DeCompressed Training of Deep Neural Networks. *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2, 4, 5, 7, 12

[58] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to imagenet? In *International Conference on Machine Learning (ICML)*, 2019. 5

[59] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 4, 8

[60] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[61] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust ImageNet models transfer better? *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 3, 4, 5, 12

[62] Victor Sanh, Thomas Wolf, and Alexander M Rush. Movement pruning: Adaptive sparsity by fine-tuning. *arXiv preprint arXiv:2005.07683*, 2020. 3

[63] Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient second-order approximation for neural network compression. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 4, 5, 7

[64] Ultralytics. Implementation of the YOLO V5 model architecture. *Available at* `https://github.com/ultralytics/yolov5`., 2021. 8

[65] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the Kronecker-factored eigenbasis. In *International Conference on Machine Learning (ICML)*, 2019. 2

[66] Jianxiong Xiao, James Hays, Krista Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 4

[67] Huanrui Yang, Wei Wen, and Hai Li. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. *arXiv preprint arXiv:1908.09979*, 2019. 2

[68] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016. 12

[69] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017. 1, 2, 4