Accurate Neural Network Pruning Requires Rethinking Sparse Optimization

Anonymous authors
Paper under double-blind review

Abstract

Obtaining versions of deep neural networks that are both highly-accurate and highly-sparse is one of the main challenges in the area of model compression, and several high-performance pruning techniques have been investigated by the community. Yet, much less is known about the interaction between sparsity and the standard stochastic optimization techniques used for training sparse networks, and most existing work uses standard dense schedules and hyperparameters for training sparse networks. In this work, we examine the impact of high sparsity on model training using the standard computer vision and natural language processing sparsity benchmarks. We begin by showing that using standard dense training recipes for sparse training is suboptimal, and provide evidence that this results in *under-training*, loosely defined as using a sub-optimal number of passes over the training data. We present training recipes for mitigating this issue for both sparse pre-training of vision models (e.g. ResNet50/ImageNet) and sparse fine-tuning of language models (e.g. BERT/GLUE), achieving state-of-the-art results in both settings in the high-sparsity regime, and providing detailed analyses for the difficulty of sparse training in both scenarios. Our work sets a new benchmark in terms of the accuracies that can be achieved under high sparsity, and should inspire further research into improving sparse model training, to reach higher accuracies under high sparsity, but also to do so efficiently.

1 Introduction

The difficulty of finding deep neural networks (DNNs) that are both *accurate and sparse*, i.e., closely match the accuracy of dense models while having a large majority of their weights set to zero, is one of the main challenges in the area of model compression. On the conceptual side, this challenge connects to fundamental questions related to the *Lottery Ticket Hypothesis (LTH)* (Frankle & Carbin, 2019; Frankle et al., 2019), which posited that such sparse masks exist, and that, in some cases, they can even allow accurate training of sparse models *from scratch*, that is, applying the sparsity mask at initialization. On the practical side, obtaining highly-sparse and accurate networks can lead to significant practical speedups, both for inference (NeuralMagic, 2022) and training (Nikdan et al., 2023).

In this work, we focus on the challenge of obtaining accurate DNNs in the high-sparsity regime, and investigate the barriers to obtaining **highly-sparse** and **highly-accurate** variants of DNNs for standard vision and language tasks. We mainly focus on two tasks that are, arguably, the standard benchmarks for sparsity in vision and language, respectively: image classification using the ResNet50 model (He et al., 2016) on the ImageNet-1K dataset (Russakovsky et al., 2015), e.g. Hoefler et al. (2021); Dong et al. (2017); Gale et al. (2019); Evci et al. (2020); Singh & Alistarh (2020); Savarese et al. (2021); Peste et al. (2021), and language modelling using the BERT-base model (Devlin et al., 2019) on the GLUE benchmark datasets (Wang et al., 2018), e.g. Sanh et al. (2020b); Hoefler et al. (2021); Kurtic & Alistarh (2022); Kurtic et al. (2022). Roughly, for both benchmarks, it is known that sparsities lower than 90% can be achieved with approximately 1% accuracy loss relative to the original dense model, but accuracy rapidly decreases in the 90-95% range (Hoefler et al., 2021; Evci et al., 2020), and that decreases are drastic at higher ($\geq 95\%$) sparsities (Singh & Alistarh, 2020; Kurtic et al., 2022). In this paper, we investigate the reasons behind this accuracy loss due to sparsity,

mainly targeting *high sparsity*, i.e. sparsities between 90% and 99%, studying the difficulty of obtaining accurate models in this range, and providing ways to circumvent it.

Contribution. We begin from the observation that, when training sparse models from scratch, following standard *dense training* schedules, *sparse models show clear evidence of undertraining*: both their accuracy and loss fail to saturate under standard number of training epochs, and their output continues to have high entropy. f This finding suggests that maximization of the accuracy of sparse models requires longer training than the dense optimization recipes adopted in most of the work on model sparsification.

Motivated by this observation, we propose a combination of techniques which can mitigate the inherent difficulty of sparse training. As a consequence, we significantly improve on the best currently-known sparsity-accuracy trade-offs on standard sparsity benchmarks for both image classification and language modelling. Specifically, we consider the two classic sparsification benchmarks in this setting: image classification (ResNet50 on ImageNet) and language modelling (BERT or SQuAD and GLUE tasks), and set new state-of-the-art results in both settings.

For image classification, we obtain, for the first time, highly-accurate sparse versions of ResNet50, such as a 90%-sparse model with 78.5% Top-1 accuracy, a 95%-sparse model with 77.7% Top-1 accuracy, and a 98%-sparse model with 75.2% Top-1 accuracy. In the same context, the highest accuracy for a dense model we could obtain is 78.78% Top-1. In addition, we show that stable results can be obtained even for extreme sparsities (e.g., 99%).

We also extend our results to language models from the BERT family Devlin et al. (2019), where we show that on challenging modeling tasks, as measured by the drop in accuracy relative to the dense model, similar techniques can improve results by 3 points in accuracy relative to the current state-of-the-art results at 90% sparsity. We arrive at these results as follows:

- We perform an analysis of the output and training characteristics of models trained using current state-of-the-art techniques, relative to their dense counterparts. First, we show that sparse DNNs obtained via many current techniques behave similarly to dense models that have been *undertrained*, i.e. executed for a sub-optimal number of epochs: specifically, they tend to have high output entropy (alternatively, low "output confidence"), which correlates with their reduced accuracy.

- This analysis provides clear evidence that optimizing *sparse models* is more difficult than standard *dense* optimization (Evci et al., 2019). This observation stands in contrast to the fact that most current sparsification techniques use standard *dense* training recipes for fine-tuning and recovery. We exploit this insight to obtain state-of-the-art accuracy for sparse models in two popular scenarios: *sparse pretraining*, i.e. training sparse models from scratch, and *sparse transfer*, i.e. optimizing a sparse pretrained model onto a target transfer task.

- In the *sparse pretraining* scenario, illustrated by the standard task of obtaining a highly-sparse ResNet50 model on the ImageNet dataset, we show that we can circumvent the difficulty of sparse training by adopting a variant of the Alternating Compressed/Decompressed (AC/DC) algorithm (Peste et al., 2021) for training sparse DNNs, which has convergence guarantees for sparse recovery. Specifically, we show that, by scaling the algorithm's runtime, we can obtain state-of-the-art results for sparse pretraining on ImageNet for ResNet50 and MobileNet models, and reach extremely high sparsities (e.g. 98% and 99%) while still obtaining stable converegence. Moreover, only sparse models benefit from extended training, whereas dense models start to overfit with longer training.

- We complement our analysis with a study of the *sparse transfer* scenario, popular in language modeling. Here, the difficulty of sparse training can manifest itself through both *undertraining* and *overfitting*, depending on the parametrization of the chosen transfer learning recipe, specifically on the training length. We address this via a modified version of the *gradual layer unfreezing* approach (Howard & Ruder, 2018), tailored towards a *sparse* transfer learning scenario, which allows us to obtain state-of-the-art results in the case of BERT-base transfer on downstream datasets.

Discussion. Overall, our results suggest that the difficulty of obtaining highly-accurate sparse models is closely linked to the difficulty of accurate sparse optimization using current state-of-the-art techniques. Specifically, our work improves the best known results on standard sparsity benchmarks, for both sparse pretraining and sparse finetuning, both in terms of absolute accuracy, and accuracy loss relative to the dense baseline. Moreover, we observe the following:

- Achieving state-of-the-art sparsity-vs-accuracy trade-offs currently requires using significant additional computational complexity and more epochs for training the sparse models, relative to the best known dense training methods. In turn, this suggests that sparse optimization may be inherently harder than its dense counterpart.
- Reaching high validation accuracy for sparse models is strongly linked to reaching low training loss, which occurs at a slower rate for sparse models in the case of SGD-based optimization. At the same time, we do observe overfitting behavior (decrease of validation accuracy w.r.t. increased training time), especially at lower sparsities.
- To further investigate the hardness of sparse optimization, we perform an analysis of the loss landscape of accurate sparse networks both in terms of sharpness and loss interpolation / mode connectivity. We observe that achieving highly-accurate sparse networks from initialization requires overcoming multiple loss barriers, and that sparsity mask exploration may be a key ingredient for overcoming these barriers.
- In addition, we investigate the relationship between standard hyperparameters such as weight decay, on the one hand, and sparsity structure, on the other. We find that careful setting of weight decay is critical for accurate sparsity, and that weight decay additionally induces (partial) structured sparsity in highly-sparse models. This provides a first explanation to the emergence of structured sparsity in unstructured sparse networks, which has been observed previously (Peste et al., 2021; Iofinova et al., 2022; Yin et al., 2023).

In summary, our results set new accuracy thresholds for sparse models using relatively simple techniques, and can be reproduced in reasonable time on commodity hardware. As such, they should serve as motivation for the community to investigate improved *sparsity-aware* optimization techniques, specifically allowing for faster, more efficient accuracy recovery.

2 Background and Motivation

Formally, accurate pruning is a constrained optimization problem which, given the objective of minimizing a loss function $\mathcal{L}$, aims to find an "optimal" sparsity mask $\mathbf{M}^\star$ with a given target sparsity s, fraction of zero parameters,[1] and weights $\mathbf{W}^\star$ such that

$$\mathbf{M}^\star, \mathbf{W}^\star = \text{argmin}_{\text{mask } \mathbf{M},\ \text{weights } \mathbf{W}} \left[\mathcal{L}(\mathbf{M} \odot \mathbf{W})\right] \quad \text{with } \text{nnz}(\mathbf{M}) \leq (1-s)\text{numel}(\mathbf{M}). \tag{1}$$

In its general form, where both the optimal mask and the optimal weights must be determined, this question is NP-hard (Blumensath & Davies, 2008), even for simple least-squares loss. However, this problem can be made tractable if we assume a fixed mask, or we wish to approximate the sparsity of the mask, e.g. Axiotis & Sviridenko (2020).

In the context of pruning, this procedure can be logically split into 1) determining the sparsity mask $\mathbf{M}$, which is often separated from 2) the optimization procedure over the non-zero weights. For instance, the standard Lottery Ticket Hypothesis (LTH) approach (Frankle & Carbin, 2019; Chen et al., 2021b) is to first identify a "ticket" mask by performing weight selection by magnitude over an already-trained model, followed by SGD-based finetuning, using the initialization and the same set of hyperparameters as for dense training.

[1] A *sparsity mask* is simply a binary tensor of the same dimensions as the model, with 0 at the indices of the sparsified entries, and 1 at the other indices.

While several novel ways of choosing or updating the sparsity mask choice (step 1), have been investigated, by and large, for the second step, that of optimizing the remaining weights, sparse training methods largely emulate the hyperparameters of the baseline dense model, including the total number of training epochs (Gale et al., 2019; Jayakumar et al., 2020; Evci et al., 2020; Peste et al., 2021). However, it is intuitive that the problem of simultaneously finding near-optimal weights and a near-optimal mask may be harder to solve than a standard dense loss minimization problem.

This naturally motivates an in-depth investigation into the following questions: *can optimization over sparse networks converge with the same rate as over dense ones?*, and *are dense training recipes well-suited for sparse training?* In this paper, we provide evidence that the answer to both questions is *negative*, suggesting that improved optimizers may be required for obtaining accurate sparse models under reduced training budgets.

3 Related Work

The goal of most sparsification methods (Hoefler et al., 2021) is to create a DNN that is as accurate as possible, while maximizing sparsity. This goal can be achieved via different strategies: for instance, *post-training sparsification methods* assume a *pretrained dense model*, from which weights are removed either in a single step (one-shot) or progressively (gradual pruning). By contrast, in *sparse training methods*, parameters are pruned from the model during training from scratch, either close to initialization (Evci et al., 2020; Jayakumar et al., 2021; Lee et al., 2019; Vanholder, 2017; Schwarz et al., 2021), or progressively as the model is trained (Han et al., 2015; Gale et al., 2019; Savarese et al., 2021). A subset of sparse training methods are *dynamic*, in the sense that weights may be reintroduced during training (Evci et al., 2020; Peste et al., 2021).

In this work, we mainly focus on the *high-sparsity regime*, in which *sparse training* methods provide the best known accuracy-vs-sparsity trade-offs. We begin by discussing methods for computer vision. Here, Gradual Magnitude Pruning (GMP), in which the lowest-magnitude weights are progressively removed throughout training, is a common baseline. In Gale et al. (2019), GMP was shown to be competitive with more sophisticated pruning methods on image classification models when properly tuned; similar results were later shown for language models (Kurtic & Alistarh, 2022).

The RigL pruning method (Evci et al., 2020) is a common, high-performing benchmark for dynamic sparse training. In this method, the weights are initially pruned to the target sparsity and trained through (sparse) stochastic gradient descent. Periodically, however, the mask is updated by selecting weights with the highest magnitude gradient, subject to a limit on the total mask change. The authors run this method using two sparsity targets - Uniform sparsity, where all layers (except the first and last) are pruned to the same proportion, and Erdős–Rényi Kernel (ERK), where layer sparsity targets are set to optimize performance. The authors test their method in the normal-schedule (100 epochs on Imagenet) and 5x training regime, getting results of 73.22% validation accuracy and 74.63% validation accuracy at 95% global (ERK) and uniform sparsity, respectively when training for 500 epochs. Extending training to 10 000 epochs (100x) further allowed the authors to produce 99% sparse (ERK) ResNet50 models with 68.5% accuracy on ImageNet. RigL was improved by combining it with ITOP (Liu et al., 2021), by altering training hyperparameters to encourage mask exploration, which was shown to improve RigL results at medium (80-90%) sparsity (see Table 1).

The GraNet(Liu et al.) method extends this approach by making it gradual - either starting from a dense network and performing RigL-like updates while simultaneously increasing sparsity until the target sparsity is achieved, or by starting by a partially sparse (50%) network and doing the same. Models trained with the sparse-init version of GraNet achieved 72.3% validation accuracy at 95% global sparsity when training for 100 epochs.

The AC/DC pruning method (Peste et al., 2021) alternates dense and sparse pruning phases of several epochs each, effectively co-training dense and sparse models. Similar to RigL, AC/DC was tested in the normal and extended training regime, creating 95% globally sparse ImageNet-1K ResNet50 models with 73.14% top-1 accuracy, and 68.44% top-1 accuracy 98% sparse models after 100 epochs of training. The authors also experiment with extended training times, producing 95% uniform sparsity ResNet50 models with 74.3% validation accuracy.

Another successful pruning approach is the combination of Powerpropagation (Schwarz et al., 2021) with Top-KAST (Jayakumar et al., 2021). In Powerpropagation, the weights are reparametrized using $f(w) = w|w|^{\alpha-1}$ for $\alpha > 1$, effectively encouraging high-magnitude weights to continue increasing while lower-magnitude weights are driven toward 0. Top-KAST is a dynamic sparse training scheme that is largely similar to RigL: in Top-KAST, for a target density D, the gradients of the top $D' < D$ weights are computed in each backpropagation round and allowed to accumulate, and the masks at these respective sparsities are periodically recomputed. The combination of these two methods results in 77.16% accuracy at 90% sparsity when trained for 3x their baseline of 32K steps.

The recently-proposed ST-3 method (Vanderschueren & Vleeschouwer, 2023) uses the technique of soft thresholding with straight-through gradient estimation to progressively prune neural networks while allowing weights to move more smoothly between the dense and sparse states. Using this method, the authors were able to achieve ImageNet accuracies of between 74% and 75% at 96% sparsity on ResNet-50, depending on the method variant used.

Additionally, some works have explored the difficulty of sparse optimization (Evci et al., 2019), explored changes to dense training pipelines to improve sparse training (ab Tessera et al., 2021; Jaiswal et al., 2022), or focused on the creation of sparse accurate neural networks outside of the standard paradigm of simultaneously searching for the optimal mask and weights. Notably, (Liu et al., 2021) explored the impact of mask exploration (that is, the total number of explored parameters at any point in sparse training), demonstrating the positive effect of extended training on both sparse network performance and total number of explored parameters. The STEP (Lu et al., 2023) learning method explored the interaction of sparsity with the Adam optimizer (Kingma & Ba, 2015), finding that the masked weights lead to an incorrect estimate of the second moment during optimization; these observations led to their proposal of a new method for N:M sparsity that alleviates these effects. The GradMax method (Evci et al., 2022) initializes a small neural network, then uses predicted gradients to grow a larger (while still small) neural network by adding additional neurons.

The problem of the sparse optimization also emerges in the context of the optimal transport (Peyré & Cuturi, 2020; Cuturi, 2013). It is often desirable to have a sparse assignment between the source and target domain. Several works have studied this question with applications to color transfer (Blondel et al., 2018) and sparse mixture of experts (Liu et al., 2022).

Language models For language models, the standard compression pipeline consists of two stages: pre-training on a large unlabeled text corpus followed by fine-tuning on a small and labeled task-specific dataset. The former is used to capture the statistical patterns and relationships that exist in the natural language, allowing the model to recognize and even generate various linguistic patterns. The latter stage, fine-tuning on a downstream task, builds on top of the learned representations and adapts them to solve specific tasks such as text classification, sentiment analysis, duplicate detection, etc. Sparsity has been explored in both stages: pruning during pre-training and pruning during fine-tuning.

Methods such as Movement Pruning (Sanh et al., 2020a) and The Optimal BERT Surgeon (oBERT) (Kurtic et al., 2022) make use of first-order (gradient) and second-order (curvature) information, respectively, to guide pruning decisions during the fine-tuning stage. However, recent work observed two problems with this approach when applied on small datasets: (Zhang et al., 2022) demonstrated instability due to large variability of estimated importance scores, while (Huang et al., 2021) observed overfitting despite reduced expressive power due to pruning. From the practical side, this approach is less favorable for practitioners as it requires extensive pruning-domain knowledge to properly configure pruners for each model and dataset combination. Therefore, the main focus of our work is on the other stage, leveraging already sparse pre-trained models with transfer learning to obtain highly accurate task-specific fine-tuned models. Prune Once for All (Prune OFA) (Zafrir et al., 2021) and oBERT (Kurtic et al., 2022) represent the most recent state-of-the-art techniques addressing this problem. Both methods first prune the model during the pre-training stage, and then apply transfer learning with a fixed sparsity mask to obtain fine-tuned and sparse models on various downstream datasets.

Impact of sparsification beyond top-1 accuracy An open area of research is the impact that pruning in general, and the choice of pruning method in particular, have on the resulting model. In particular, pruned

models have been shown to be more vulnerable to bias (Hooker et al., 2019; 2020; Iofinova et al., 2023), and worse at prediction accuracy under distribution shift (Liebenwein et al., 2021). Recent works by (Chen et al., 2021a) and (Iofinova et al., 2023) investigate the effects of pruning on a range of model trustworthiness metrics and find mixed results, with sparse neural networks having better calibration, but exaggerating spurious patterns in the existing data. Finally, works such as (Iofinova et al., 2022) and (Chen et al., 2021b) investigated the capacity of sparse CNNs for domain adaptation via transfer learning, finding that sparsely trained networks can have more generalizable features than dense ones.

4 The Difficulty of Sparse Pretraining of Vision Models

4.1 Sparse Vision Models Show Evidence of "Undertraining"

We begin by investigating correlations between the performance and output characteristics of dense and sparse models trained for increasing number of epochs. Specifically, we examine two key metrics: *Top-1 accuracy* on the validation/test set, and the *loss on the train set* for the trained models, while scaling the number of training epochs and the associated hyperparameters correspondingly.

We will examine the evolution of these metrics as we increase the number of epochs, in parallel for sparse and dense models. We specifically look out for instances where sparse models behave similar to dense ones that have been trained for a sub-optimal (too low) number of epochs, a phenomenon we simply call *undertraining*.

Metrics: Output Loss and Entropy. We examine model fit to the training data via the training loss at the last epoch of training. For multiclass classification, traditionally cross-entropy loss is used. We compute the cross-entropy loss by taking the softmax over the vector of output values of the network and then applying the standard cross-entropy formula, where the cross-entropy is taken with respect to the correct label distribution for the model (1 for the correct class and 0 otherwise). For an output of a network outputting a vector $Z = (z_1, z_2, ..., z_C)$ of size C with correct label L, cross-entropy CE is given by the following formula:

$$CE(Z) = -\log\left(\frac{e^{z_L}}{\sum_{j=1}^{C} e^{z_j}}\right). \tag{2}$$

Intuitively, the loss of the model is related to its "confidence" in the correct predictions, or equivalently could be said to measure the model's fit to the training data.

We use this quantity as it is conventional with respect to measuring model convergence; however, we consider the *entropy* computed over *test* data to be an equally good choice, as it captures the model's confidence in its predictions (whether they be correct or incorrect) and can be computed on a *test* set, without access to the correct labels. We show in Appendix C that the two metrics give nearly identical results in our experiments.

We expect a sufficiently large and well-trained model to have low loss on the training data. However, as is conventionally known, continued training on dense and low-sparsity models can result in in overfitting will lower these metrics further. Here we investigate whether the same rule applies to models with higher sparsity.

Experimental setup. We examine validation accuracy on trained sparse and dense ResNet50 models on the ImageNet-1K dataset and compare it to the train loss on the last epoch of training. All models were trained using standard hyperparameters (see Appendix A) except for the difference in number of training of epochs in different experiments. Measurements represent the final accuracy and training loss after the last training epoch, so each marker on the plots represents a full experiment, rather than an intermediate checkpoint. Sparse models were pruned with Alternating Compression/Decompression (AC/DC) (Peste et al., 2021), likewise adjusting the total number of compressed and decompressed phases to the total run length. AC/DC was chosen as it was among the best-performing methods across all sparsities and training lengths (see Section 4.2.1). We use the FFCV library (Leclerc et al., 2022) for fast loading of the data. In contrast with other runs presented in this paper, we do not use progressive resizing or label smoothing, as the latter

explicitly encourages high prediction entropy and cross-entropy. In these experiments, we keep the first and last layer dense.

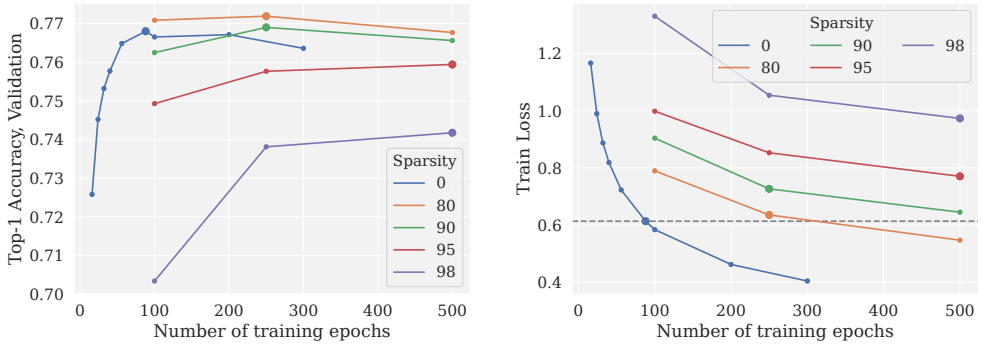

Figure 1: Average validation accuracy (left), and Train loss at final epoch (right) for sparse and dense ImageNet models trained for different numbers of epochs. The highest-accuracy model for each sparsity level is highlighted with a larger marker. The cross-entropy loss and entropy level of the dense model is also shown with a dashed line, to simplify comparison.

Results. Our results are presented in Figure 1. On the left panel, we show the top-1 accuracy of the final models. We observe that 80% and 90% sparse models reach an accuracy that is similar to dense models, even slightly exceeding dense accuracy at 80% sparsity. Accuracy drops at higher sparsity (95% and 98%); this is consistent with the original AC/DC paper and results from other pruning methods. Examining accuracy across epoch budgets, and focusing on the best-performing model for each sparsity level, we observe the following:

- *The dense model requires the fewest epochs* (88) to reach its best validation accuracy, and extending the training recipe results in *worse performance for the dense model*, commonly known as "overfitting."

- *The outcome changes if we examine sparse models*, for which the ideal training length increases with sparsity: 250 epochs for 80% and 90% sparse models, and at least 500 epochs—the longest schedule we tried in this experiment—for 95% and 98% sparse models. Even at 500 epochs, the accuracy increase/loss decrease for these models does not appear to be saturated.

We now examine loss on the training dataset in more detail. We observe that the training loss always decreases when the number of training epochs is increased. However, sparse models trained for the standard 100 epochs show similar training loss to dense models trained for far fewer epochs. For example, dense models trained for 24 epochs have a similar training loss to 95% sparse models trained for 100 epochs, while dense models trained for 100 epochs have a slightly lower training loss than 80%-sparse models trained for 250 epochs. When we consider the best-performing models at their respective sparsity levels, we find that they have similar training loss to the top-performing dense model, in cases where such low loss/entropy can be achieved in a reasonable number of epochs (at 80% and 90% sparsity); at all sparsities. Further, continuing to train sparse models to until training loss drops below the training loss of the optimal dense model results in worse validation accuracy (overfitting).

Discussion. These findings further support our hypothesis that, due to the inherent difficulty of sparse optimization, using standard training recipes is not sufficient for sparse training, and suggests that longer training may mitigate this effect. Further, results suggest that training loss can act as a useful criterion to

validate that the sparse models are properly trained[2], with the latter criterion being also useful in cases where access to train data, or to any labeled data, is not possible.

In Appendix Section C, we consider the alternative Validation entropy metric, and present a similar validation on the Celeb-A dataset.

4.2 State-of-the-Art Accurate Sparse Pre-Training on ImageNet

The above observations for vision models suggest that successful sparse training may benefit from an extended training schedule. We now build on this idea to achieve state-of-the-art results for the classic ResNet50/ImageNet benchmark by using an extended-training version of AC/DC, which we call AC/DC++.

4.2.1 Comparing Sparse Training Methods

For the following experiments, we start from the current state-of-the-art training approach for ResNet50/ImageNet training, using the Pytorch FFCV package (Leclerc et al., 2022). In addition to an extended training schedule, we use label smoothing and a linear learning rate decay with warm-up, as well as progressive resizing of input samples [3]. In this context, we implemented three leading sparse training methods: Gradual Magnitude Pruning (GMP) (Zhu & Gupta, 2017), RigL (Evci et al., 2020) and AC/DC (Peste et al., 2021), which we execute for an increasing number of epochs between 100 (standard) and 1000 (10x). For this, we scale the original training schedule proportionally, following the proportions employed by the original methods. For this experiment, models are compressed to 80%, 90%, and 95% sparsity. Following the most common experimental setup we prune all weights in convolutional and linear layers (including input convolution and classification head). The exact training recipe is presented in detail in Appendix A. We note that each of the experiments presented in the paper takes less than a day on a standard 8-GPU NVIDIA RTX 3090 server. The results, in terms of accuracy and loss vs number of training epochs are presented in Figure 2 for 95% sparsity and in Figure P.9.

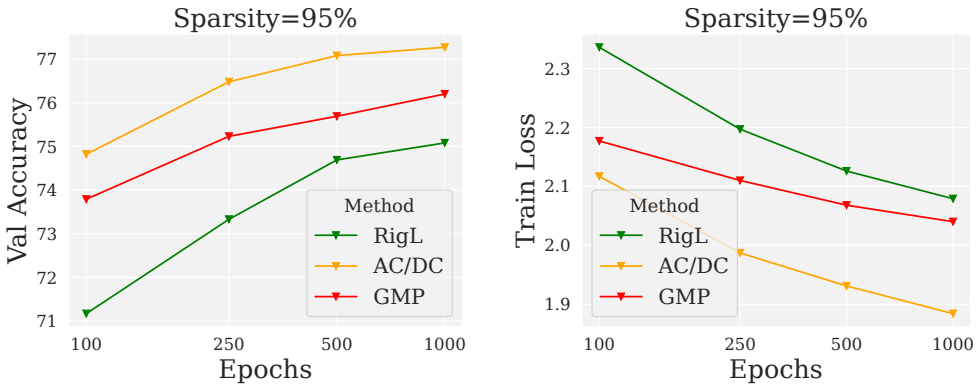

Figure 2: (**left**) Validation accuracy on ImageNet-1k vs number of epochs for different sparse training methods. (**right**) Training loss on ImageNet-1k vs number of epochs for different sparse training methods.

Results. The results show a strong correlation between how well the methods achieve reduction in loss and their validation accuracy. This reinforces the point that sparse training methods saturate slower, both in terms of training loss and validation accuracy. This has also been investigated by prior work: Gale et al. (Gale et al., 2019) found that extended training did improved results for GMP in some cases, while RigL (Evci et al., 2020) and Powerpropagation (Schwarz et al., 2021) found diminishing improvements. At the same time,

[2]The 98% sparse model will likely never reach the entropy of the optimal dense model, suggesting that the accuracy may continue to improve with very long training schedules. In fact, the authors of RigL trained a 99% sparse model for 100 times the dense training time and were not able to saturate its accuracy. See `www.github.com/google-research/rigl#extended-training-results`.

[3]We follow the setup from the FFCV ImageNet example repository for ResNet50.

we notice a significant difference between methods: specifically, AC/DC starts at a slightly better accuracy point, and consistently outperforms other methods both in terms of loss achieved, and in terms of validation accuracy, as we increase training time. (This is consistent with the AC/DC original results, executed at 100 epochs (Peste et al., 2021).) We observe that this correlates with the theoretical computational cost (FLOPs) of the methods: AC/DC will use more FLOPs than other methods due to the dense training phases, while GMP uses more FLOPs than RigL due to gradually increasing sparsity. In turn, this could also be correlated with the amount of mask exploration performed by the algorithm during training. At low sparsity RigL performs slightly better than GMP, but for higher sparsity GMP appears to perform better. For the smallest 80%, 90% AC/DC reaches a saturation point, whereas in all other setups model performance continues to improve with training budget.

4.2.2 Sparsity-vs-Accuracy Results

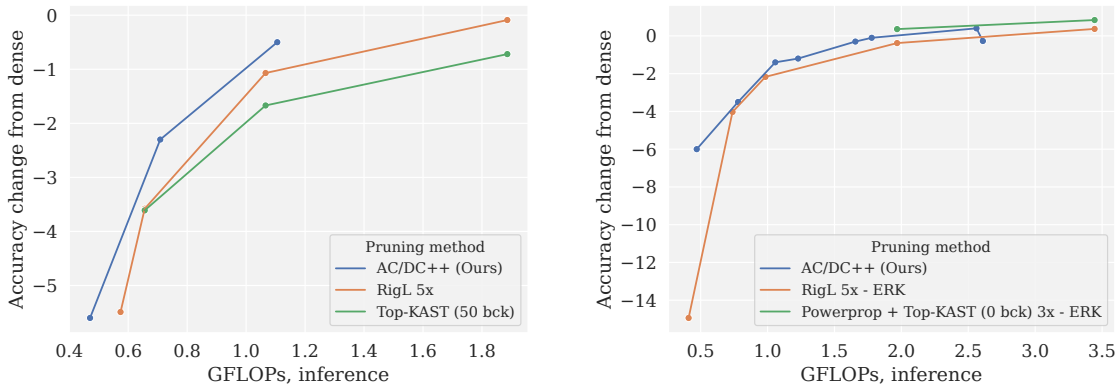

Figure 3: Comparison of Accuracy change from dense baseline as a function of Inference FLOPs for leading sparse training methods, under uniform sparsity constraints (**left**) and global sparsity constraints (**right**). Due to a lack of a standard benchmark, global and Erdős–Rényi Kernel (ERK) sparsity constraints were grouped together. Both sparsity schedules of AC/DC++ (with all layers sparsified and with the first and last layer kept dense) are plotted together.

Goals and Metrics. Based on these results, in this section, we aim to improve the best known sparsity-versus-accuracy trade-offs by performing a thorough ablation over sparsities and training length parameters. We compare our results to the highest-performing previously published sparse training methods. In particular, we compare an extended-training version of AC/DC, which we call AC/DC++, results reported in the original RigL, ST-3, and Powerpropagation papers, as well as many other existing pruning methods.[4]. All methods are described in Section 3. In cases where the authors conducted extended training using their method, we present those numbers, and we use the FLOPs-optimized ST-3^{σ} variant. AC/DC++ candidate models were trained for four preset training lengths (1x, 2.5x, 5x and 10x the standard ImageNet training time on ResNet50) at all sparsity levels, and we chose the best results obtained by ablating over the length of the training run.

As different methods have different computational budgets and different dense baselines, to ensure a fair comparison, we examine the model performance both in terms of *Top-1 Validation accuracy*, and the *Top-1 Validation accuracy difference from the corresponding dense baseline*. We use the best available numbers originally reported in the papers introducing the methods for comparisons.

Experimental Setup. We compare two pruning regimes. First, we consider *Uniform Pruning*, in which every layer is pruned exactly to the target sparsity, except for the first and last layer, which are left dense. Second, we consider the *Global/Nonuniform Pruning* regime, in which the sparsity budget is set globally.

[4]The most successful Powerpropagation approach presented in the paper combines this method with Top-KAST; we use this benchmark, as it performs better than Top-KAST alone.

Method	Top-1 accuracy (%) Dense (D)	Top-1 accuracy (%) Sparse (S)	Δ Accuracy $100 \times \frac{(S-D)}{D}$	Sparsity (%)	Remaining # of params	Inference FLOPs prop. of dense
Sparse Training						
AC/DC (Peste et al., 2021)	76.8	75.03	-1.77	90	2.56 M	0.18
GraNet($s_0 = 0.5$) (Liu et al.)	76.80	74.5	-1.3	90	-	0.20
Powerpropagation + Top-KAST FLD (Schwarz et al., 2021)	76.8	75.23	-1.57	90	-	-
Powerpropagation + Top-KAST ERK (Schwarz et al., 2021)	76.80	75.74	-1.06	90	-	0.24
RIGL ERK 1x (Evci et al., 2020)	76.80	73.00	-4.94	90	-	0.24
RIGL-ITOP ERK 1x (Liu et al., 2021)	76.80	73.82	-2.98	90	-	0.24
ST-3 (Vanderschueren & Vleeschouwer, 2023)	77.10	75.28	-1.82	90	-	0.24
STR (Kusupati et al., 2020)	77.01	74.31	-3.51	90.23	2.49 M	-
Variational Dropout (Molchanov et al., 2017)	76.69	73.84	-3.72	90.27	2.49 M	-
Post-training sparsification						
Global Magnitude (Singh & Alistarh, 2020)	77.01	75.15	-2.42	90	2.56 M	-
WoodFisher (Singh & Alistarh, 2020)	77.01	75.21	-2.34	90	2.56 M	-
Extended sparse training						
AC/DC++ 5x (this work)	78.78	78.49	-0.29	90	2.60 M	0.2
AC/DC++ FLD 5x (this work)	78.78	78.6	-0.18	90	4.45 M	0.22
GMP FLD 1.5x (Gale et al., 2019)	76.69	75.16	-1.53	90	-	-
GraNet($s_0 = 0.5$) 2.5x (Liu et al.)	76.80	76.4	-0.4	90	-	0.20
Powerpropagation+Top-KAST ERK 3x(Schwarz et al., 2021)	76.80	77.16	+0.36	90	-	0.24
RIGL ERK 5x (Evci et al., 2020)	76.80	76.42	-0.38	90	-	0.24
RIGL-ITOP ERK 5x (Liu et al., 2021)	76.80	75.50	-1.30	90	-	0.24
Sparse Training						
AC/DC (Peste et al., 2021)	76.8	73.14	-3.66	95	1.28 M	0.11
GraNet($s_0 = 0.5$) (Liu et al.)	76.80	72.3	-6.5	95	-	0.12
Powerpropagation + Top-KAST FLD(Schwarz et al., 2021)	76.8	73.25	-3.55	95	-	-
RIGL ERK 1x (Evci et al., 2020)	76.80	70.00	-8.85	95	-	0.12
ST-3 (Vanderschueren & Vleeschouwer, 2023)	77.10	74.46	-2.64	95	-	0.13
STR (Kusupati et al., 2020)	77.01	70.40	-8.58	95.03	1.27 M	-
Variational Dropout (Molchanov et al., 2017)	76.69	71.81	-6.36	94.94	1.30 M	-
Post-training sparsification						
Global Magnitude (Singh & Alistarh, 2020)	77.01	71.72	-6.29	95	1.28 M	-
WoodFisher (Singh & Alistarh, 2020)	77.01	72.12	-6.89	95	1.28 M	-
M-FAC (Frantar et al., 2021)	77.01	72.6	-4.41	95	1.28 M	-
Extended sparse training						
AC/DC++ 10x (this work)	78.78	77.27	-1.48	95	1.33 M	0.13
AC/DC++ FLD 10x (this work)	78.78	77.7	-1.08	95	3.28 M	0.14
GMP FLD 1.5x (Gale et al., 2019)	76.69	72.71	-3.98	95	1.28 M	-
RIGL ERK 5x (Evci et al., 2020)	76.80	74.63	-2.17	95	1.28 M	0.12
Sparse training						
AC/DC (Peste et al., 2021)	76.8	68.44	-9.36	98	0.7 M	0.06
ST-3 (Vanderschueren & Vleeschouwer, 2023)	77.10	70.46	-6.64	98	-	0.07
STR (Kusupati et al., 2020)	77.01	70.40	-8.58	98	-	-
Variational Dropout (Molchanov et al., 2017)	76.69	64.52	-15.87	98.57	0.36 M	-
Post-training sparsification						
M-FAC (Frantar et al., 2021)	77.01	67.5	-9.51	98	-	-
WoodFisher (Singh & Alistarh, 2020)	77.01	65.55	-11.46	98	0.51M	-
Extended sparse training						
AC/DC++ 10x (this work)	78.78	74.06	-4.72	98	0.51 M	-
AC/DC++ FLD 10x (this work)	78.78	76.6	-2.28	98	2.58 M	0.09
Sparse training						
ST-3 (Vanderschueren & Vleeschouwer, 2023)	77.10	63.88	-13.22	99	-	0.04
Extended sparse training						
AC/DC++ FLD 10x (this work)	78.78	72.7	-6.08	99	2.34 M	0.06
RIGL ERK 5x (Evci et al., 2020)	76.80	61.86	-15.94	99	-	0.05
RIGL ERK 10x (Evci et al., 2020)	76.80	63.89	-12.91	99	-	0.05
RIGL ERK 50x (Evci et al., 2020)	76.80	66.94	-9.86	99	-	0.05
RIGL ERK 100x (Evci et al., 2020)	76.80	68.15	-8.65	99	-	0.05

Table 1: Comparison between modern sparse training methods on ImageNet-1k with ResNet-50 models for various sparsity targets. ERK refers to the Erdos-Renyi Kernel sparsity distribution. FLD refers to the first and last layers being dense (AC/DC++) or the first layer being dense and the last layer being 80% sparse (GMP, PowerPropagation).

Different works apportion the global budget differently, and also differ with respect to which parts of the network are subject to the global constraint. In particular, Extended GMP (Gale et al., 2019) and Top-KAST do not prune the first layer, prune the last layer to a fixed 80% sparsity, and prune the other layers using a global magnitude criterion. RigL uses an Erdős–Rényi-Kernel distribution for layer sparsity targets, and leaves only the first layer dense. The original AC/DC work uses global sparsity and prunes all convolutional and FC layers. Therefore, to create a more fair comparison, we consider estimated Floating-Point Operations (FLOPs) necessary for inference; these are computed as in (Evci et al., 2020). Using FLOPs also equalizes methods across slight variations in ResNet50 architectures, and so we use it also for the Uniform pruning comparison. In addition, we use two pruning schedules for AC/DC++: one which leaves the first and last layer dense and prunes the remaining layers using a global magnitude criterion, and one that prunes all layers

using the global magnitude criterion. We do not ablate between the two, but rather present both sets of results in Figure 3 (jointly) and Table 1 (separately).

We emphasize two key points regarding our comparisons:

1. Looking at accuracy alone favors AC/DC++, as it has a higher dense baseline: since we use several recent training innovations, the dense model can reach 78.78% dense accuracy over 100 epochs. Therefore, it becomes more challenging to maintain the performance of the dense model for highly sparse model compared to less-optimized baseline.

2. This is why we also examine *accuracy difference relative to the dense baseline*: this favors other methods, as they are benchmarked against a standard-recipe model that reaches lower 76.8% accuracy (77.1% for ST-3).

Results. The results are presented in Figure 3 and Table 1. We observe that, for uniform pruning budgets, the AC/DC++ models outperform other methods, both in terms of absolute and relative validation accuracy. This is true even when we consider extended-training schedules for other methods, although we believe we are the first to systematically investigate the impact of increasing training schedules at these sparsity levels.[5] When looking at models trained with global pruning budgets, we observe that AC/DC++ obtains the highest absolute validation accuracy, compared to results reported previously in literature. When considering accuracy change from the dense line, AC-DC++ loses lefss accuracy than other methods at very high sparsities (lowest FLOPs), despite having the highest-performing dense baseline; at lower sparsity (90%), it is competitive with other extended training methods.

4.3 Additional validations and ablations

We performed additional analysis and ablations, to validate the performance and hyperparameters of the AC/DC++ model, and better understand the factors contributing to its high performance. These studies are summarized briefly below, and available in the Appendix.

4.3.1 Additional Evaluations

Additional quality evaluations. Having demonstrated that extended training has a strong positive effect on sparse model top-1 test accuracy, we further investigate the impact of extended training on other aspects of model quality. We consider two additional quality metrics: their performance in transfer learning scenarios and robustness to common image perturbations.

Iofinova et al. (2022) demonstrated that equally sparse models with comparable performance on the original task can vary widely in their performance once finetuned on other, smaller transfer tasks. We compare the transfer performance of dense and 95% sparse AC/DC++ models, both trained for 100 and 1000 epochs in two transfer learning regimes: linear finetuning, where the hidden layers of the model are trained only on the larger (ImageNet) task, and only the final FC layer is trained on the transfer task, and full-network finetuning, where all layers are finetuned on the transfer task. We find that extended training improves the transfer performance for both transfer scenarios for 95% sparse models, but is largely neutral for dense models. Full details of the experiment and evaluation are given in Appendix E.

We test robustness by measuring model performance on the ImageNet-C dataset (Hendrycks & Dietterich, 2019), which digitally adds 19 types of perturbations to the ImageNet-1K validation set. (Liebenwein et al., 2021) and (Hooker et al., 2019) have found that compressed models are less robust under many types of perturbations, compared to dense models. As before, we consider dense and 95% sparse AC/DC++ models trained for 100–1000 total epochs. We find that robustness to perturbations increases with training time for sparse models, but stays the same for dense ones. Full details of the experiment and evaluation are given in Appendix F.

[5] In prior work, RigL executed >5x extended training for a 99%-sparse model only (Evci et al., 2020).

Additional models. In Appendix D,we show that extended training with AC/DC++ produces state-of-the-art results on the MobileNet-V1 architecture as well.

4.3.2 Parameter Ablations

AC/DC dense fraction. We note that AC/DC is relatively expensive in terms of training FLOPs as compared to other sparse training methods. In Appendices H and L we investigate if this can be improved by shortening the duration of the decompression phase relative to the compression phase, and by using a lower, but nonzero sparsity during the decompression phases of AC/DC, respectively. We find that, for models with a target sparsity of 95%, spending 50% of the training time in each phase is optimal, consistent with the original model recipe. However, shortening the decompression phase so that the model spends only 20% of the training time in that phase has a small 0.2% accuracy drop. Additionally, we find that, for 95% sparse models, decompression phases that are up to 70% sparse have matching or better performance to the original AC/DC models with 0% sparse decompression phases. Therefore, if minimizing floating-point operations during training is an objective, it is possible to make the training more efficient in that regard.

AC/DC phase duration. In Appendix I we confirm that for ResNet50 models trained on ImageNet and assuming equally-sized compression and decompression phases, the 5-epoch phase duration used in the initial paper is optimal.

Impact of weight decay. in Appendix L, we consider the impact of weight decay on AC/DC model performance and sparsity. We find that using high values of weight decay (1e-3) results in models that stay largely sparse even during the decompression phase, and that are considerably less accurate. Conversely, using low values of weight decay (1e-5 and 1e-6) result in models with very low sparsity during the decompression phase, and also decreased performance.

4.3.3 Additional Analyses

Mask Analysis. We investigate the importance of mask updates during sparse training. Specifically, we examine how much masks change as training progresses. We find (see Figure 4 (left)) that for both RigL and AC/DC, masks change substantially more early in the training, with an Intersection/Union scores of 0.3-0.4 for AC/DC 95% sparse models and 0.92-0.95 for RigL during the first 20% of training steps (recall that RigL masks are updated far more frequently than AC/DC masks). Later in the training, masks stabilize, with less change in consecutive masks. Full results are provided in Appendix J,

Loss landscape analysis. In Appendix M, we investigate the sharpness of the loss landscape (as measured by an approximation to the highest eigenvalue of the Hessian matrix at the point of convergence). We find (see Figure 4 (left)) that, across all methods, sharpness increases with the length of the training run, indicating that sharper minima require extended training to be reached via SGD. Additionally, sharpness decreases with the increase of sparsity. All sparse training methods attain lower sharpness compared to the dense model.

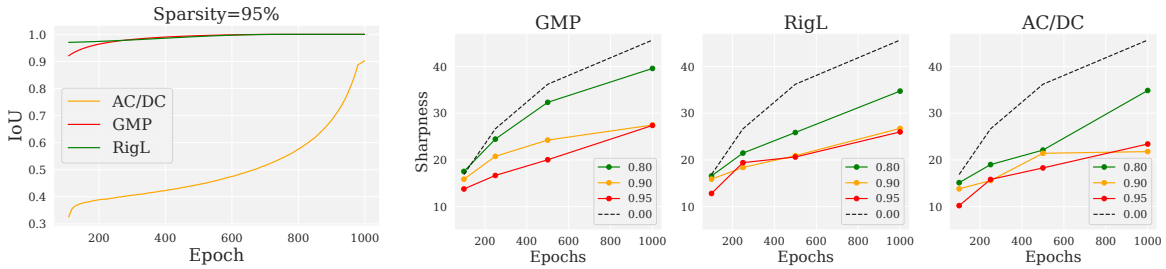

Figure 4: (**left**) Mask IoU between two consecutive checkpoints at 95% sparsity. (**right**) Sharpness (highest eigenvalue) of the loss surface vs number of epochs. Dashed lines correspond to the dense model.

Structured sparsity. In Appendix K, we measure the number of channels that have been completely zeroed out during (unstructured) sparse training. We find that AC/DC and AC/DC++ models have a considerable amount of sparse channels, with AC/DC++ models having considerably higher structured sparsity than AC/DC for every sparsity.

Comparison with smaller dense model. In Appendix G we confirm that a 95% sparse ResNet50 model substantially outperforms a half-width dense ResNet50 model trained for the same budget.

Different sparsity patterns. In Appendix N, we confirm that adding constraints on the sparsity pattern (such as uniform per-layer sparsity and block-4 sparsity) lowers model accuracy.

5 The Difficulty of Sparse Transfer in Language Modelling

Next, we extend the analysis to language models, specifically to the very common scenario in which a large language model (BERT-base) is adapted to a specific task via *finetuning*. Thus, here we will examine the impact of sparsity target, number of iterations, loss function, and hyper-paramterization on the optimal recipe for the task of *finetuning* sparse models on the downstream dataset.

In the context of our study, this setup naturally leads to the following questions: *"do finetuned sparse language models suffer from being undertrained on the downstream task?"*, and *"if yes, does the simple recipe of extended training suffice to mitigate the issue?"*. In this section, we will show that when dense finetuning recipes are used for sparse transfer learning in language models, the resulting models are indeed undertrained and have poor transfer performance. However, we also note an additional difficulty: extended training does not suffice to mitigate the issue, because sparse language models quickly shift from being undertrained to an overfitting regime. The latter is a far larger problem in language understanding tasks than in visual ones, which is likely why we don't observe the same issues with visual transfer learning in Appendix E - there we simply use a long finetuning schedule in all cases. In this section, we explore the problem of balancing under- and over-training in sparse language models and propose a sparse finetuning recipe for creating properly tuned sparse models.

5.1 Under Standard Dense Transfer Learning Recipes, Sparse Models are Undertrained

Experimental Setup. In our experiments, we make use of open-sourced *sparse* pre-trained BERT-base models obtained by (Kurtic et al., 2022). On top of these, we apply various transfer learning recipes to obtain fine-tuned sparse models on datasets from the popular GLUE benchmark (Wang et al., 2018). For fair comparisons with results from prior work, we employ early stopping for all methods. We provide more details about each dataset in Appendix O.

The most popular and widely adopted dense transfer learning recipe consists of fine-tuning all weights with linearly decaying learning rate for as much as two or three epochs on the target downstream task. In Table 2 we present results obtained with this approach when applied to sparse models, and denote it as a *dense-transfer recipe*. Under the same transfer learning recipe, we clearly observe significant gaps (up to 14 accuracy points on RTE and CoLA) between the transfer accuracy of the *dense model* (*Dense BERT-base*), and the transfer accuracy of the *sparse model* (*Dense-transfer recipe*).

5.2 Extended Training Shifts from Undertraining to Overfitting

Observing that the dense transfer learning recipe does not produce competitive sparse finetuned models, we attempt to scale the length of the recipe to mitigate undertraining. Surprisingly, for sparse language models, this simple technique does not yield a unique setup with consistently better results as models quickly shift from undertraining to an overfitting regime, in which training loss goes to zero, while validation accuracy decreases sharply. To demonstrate this overfitting effect with the extended recipe, in Table 2, we compare results obtained with this approach (*Extended dense-transfer recipe*) against doing a full sweep of finetuning runs with rescaled recipes to #epochs $\in \{1, 2, 3, ..., \text{extended} - 1\}$ (*Full sweep of rescaled recipes*).

Table 2: Sparse-transfer performance of 90% sparse pre-trained BERT-base model on the dev-set of the corresponding GLUE task, obtained with dense and extended dense (#epochs=8) transfer learning recipes, as well as with the full sweep of rescaled recipes (#epochs $\in \{1, 2, ..., 7\}$).

Sparse-transfer	RTE Acc	QNLI Acc	MRPC Acc	SST-2 Acc	CoLA Mcc	STS-B Pear	MNLI Acc	QQP Acc
Dense BERT-base (baseline)	66.1	91.3	85.5	93.0	56.8	88.9	84.6	91.5
Dense-transfer recipe	52.4	88.9	82.8	91.2	42.5	87.1	**82.2**	90.0
Extended dense-transfer recipe	55.2	88.7	**85.6**	91.4	47.2	87.6	81.6	90.3
Full sweep of rescaled recipes	**57.0**	**89.3**	84.1	**92.0**	**48.5**	**88.0**	**82.2**	**90.4**
Best recipe length	5 ep	2 ep	5 ep	2 ep	7 ep	4 ep	3 ep	5 ep

The results suggest that with the existing recipes, there is no one-size-fits-all solution. Versions of this rescaling approach have been utilized by prior works like (Kurtic et al., 2022) and (Zafrir et al., 2021) to obtain accurate sparse models on various downstream datasets. However, this approach comes with a huge computational burden: for each rescaled recipe, a full hyperparameter sweep over relevant parameters has to be done in order to obtain competitive finetuned sparse models. Due to practicality and associated costs, this is not a desirable solution in practice.

5.3 Sparse Transfer Learning for Language Models

In the previous section, we have demonstrated the following three problems with the existing approach of either using the dense finetuning recipe, or simply extending it for sparse finetuning:

1. following dense-transfer recipes, sparse language models are undertrained;
2. even at high sparsities, these models can still exhibit overfitting behavior under the extended training regime;
3. finding the optimal recipe to mitigate undertraining and overfitting has major computational burdens.

To address these issues, we propose a simple approach for sparse transfer in NLP, which produces highly accurate and competitive sparse models on a wide range of downstream datasets with minimal hyperparameter tuning. Our technique is inspired by the idea of gradual layer unfreezing presented in the ULMFiT framework (Howard & Ruder, 2018), which introduced a universal framework for fine-tuning *dense* language models for text-classification tasks, with a focus on LSTM models (Merity et al., 2017). Based on ULMFiT and findings of (Yosinski et al., 2014), which suggests that different layers capture different information and therefore should be fine-tuned to different extents, we adopt the idea of gradual unfreezing and adjust it for *transformer-based (Vaswani et al., 2017) sparse* language models.

More specifically, we focus on the popular BERT-base model which consists of three groups of layers: embeddings, 12 identical transformer blocks, and a task-specific classifier head. Sparsified versions of this model, which are the main interest of this work, prune all linear layers across all transformer blocks, which is the standard practice in literature (Sanh et al., 2020b; Kurtic & Alistarh, 2022; Kurtic et al., 2022; Zafrir et al., 2021) and brings the best accuracy-vs-latency trade-offs (Kurtic et al., 2022).

Our approach can be summarized as follows. For each downstream task, we start from a sparse pre-trained model produced by (Kurtic et al., 2022) and randomly initialize a task-specific classifier head. Then we freeze all embeddings and sparsified linear weights, while keeping their biases and corresponding LayerNorm (Ba et al., 2016) layers unfrozen and trainable. We start by finetuning only the classifier head and all other trainable parameters (biases and LayerNorms) for one epoch, and then follow the same process from back-to-front by unfreezing the unpruned linear weights in preceding transformer blocks. After the last layer is unfrozen and finetuned, we continue finetuning all layers together for one more epoch.

Given that at each epoch we have a different model architecture (one more sparse transformer block unfrozen relative to the previous epoch), we finetune it with the linearly decaying learning rate and then rewind back

Table 3: Our sparse-transfer performance of 90% sparse pre-trained BERT-base model on the dev-set of the corresponding GLUE tasks, benchmarked against the current state-of-the-art sparse-transfer results from Prune OFA (Zafrir et al., 2021) and oBERT (Kurtic et al., 2022).

Sparse-transfer	RTE Acc	QNLI Acc	MRPC F1 / Acc	SST-2 Acc	CoLA Mcc	STS-B Pear / Spear	MNLI m / mm	QQP Acc / F1
Dense BERT-base	66.1	91.3	89.8 / 85.5	93.0	56.8	88.9 / 88.5	84.6 / 83.4	91.5 / 88.5
Prune OFA (Zafrir et al., 2021)	N/A	89.1	N/A	90.9	N/A	N/A	81.5 / 82.4	**90.9** / **87.6**
oBERT (Kurtic et al., 2022)	57.0	89.3	89.3 / **85.6**	**92.0**	48.5	**88.0** / **87.6**	82.2 / 82.5	90.4 / 87.1
This work	**60.1**	**90.5**	**89.7** / 85.2	91.8	**51.4**	87.2 / 87.1	**83.7** / **83.8**	**90.9** / **87.6**

Figure 5: Evaluation loss (lower is better) and F1 score (higher is better) during sparse-transfer with oBERT (Kurtic et al., 2022) and our approach on MRPC dataset.

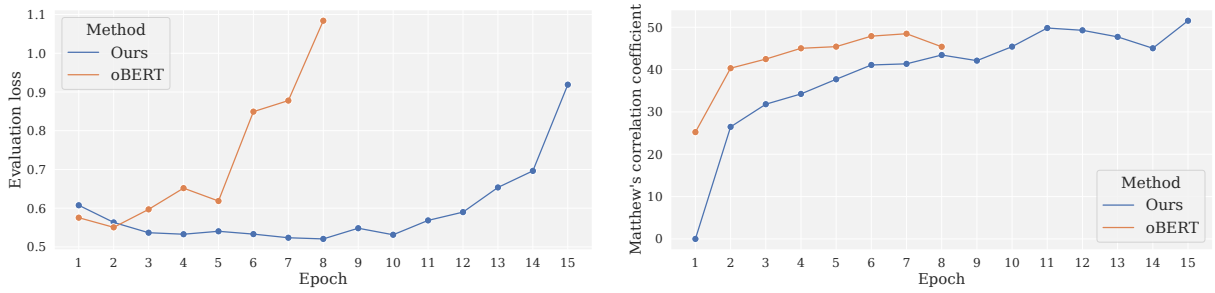

Figure 6: Evaluation loss (lower is better) and Matthew's correlation coefficient (higher is better) during sparse-transfer with oBERT (Kurtic et al., 2022) and our approach on CoLA dataset.

to the initial value for the next epoch. We have also tried the slanted triangular learning rate schedule proposed in ULMFiT, but we found the warmup phase not very helpful as it is known that sparse language models usually require much higher learning rates relative to their dense counterparts in order to train and converge successfully (Kurtic & Alistarh, 2022).

To validate the effectiveness of our proposed sparse transfer approach, we benchmark it against the two current state-of-the-art sparse-transfer results presented in *Prune Once for All (Prune OFA)* (Zafrir et al., 2021) and *The Optimal BERT Surgeon (oBERT)* (Kurtic et al., 2022) papers. The former makes use of knowledge distillation from a finetuned dense teacher model, while the latter uses a full sweep over extended and rescaled dense transfer recipes, such as the ones we presented in Section 5.2. As can be seen from Table 3, our approach outperforms highly competitive results by Prune OFA in all, and oBERT in eight out of twelve datasets, setting new state-of-the-art accuracy-vs-sparsity results for many tasks in the GLUE benchmark suite. It is worth emphasizing that all of our results are obtained with significantly less hyperparameter tuning than the other two competing methods, which aligns with our goal of finding a stable one-size-fits-all solution for the sparse-transfer problem. We search and tune the initial learning rate in {1e-4, 2e-4, 3e-4}, and dropout in {0.05, 0.1}, and report mean performance over the two best runs. Thus, our grid consists of only 6 different combinations for each considered dataset, whereas competing approaches sweep over 54 (Zafrir et al.,

2021) and 24 (Kurtic et al., 2022) different combinations. It is worth emphasizing that all of the considered methods, including ours, have noticeable variability in results on small datasets across different seeds and hyperparameter configurations, which aligns with findings of (Devlin et al., 2019).

To better understand what happens during our proposed sparse transfer learning setup, and to develop an intuition about why it is able to provide stable and competitive results across many different datasets ranging in sizes from 2.4k (RTE) and 392k (MNLI) labeled samples, we visualize evaluation loss and evaluation accuracy metrics over the entire transfer learning process in Figures 5 and 6. As can be seen, our approach enables slower and therefore more stable transfer learning on the target datasets which effectively prevents overfitting, even though the total number of epochs is two times larger than the extended dense-transfer recipes analyzed in Section 5.2. This aligns with findings in ULMFiT, which demonstrates that gradual unfreezing in combination with a carefully designed learning rate schedule prevents catastrophic forgetting and enables robust transfer learning across a wide range of different downstream tasks.

6 Conclusion

In this work, we examined the impact of high sparsity on model training under standard computer vision and natural language recognition scenarios, and provided evidence that traditional training recipes used for dense models are generally too short for sparse training. Starting from this observation, we were able to produce state-of-the-art models for sparse computer vision on two classic benchmarks for pruning: the ResNet50/ImageNet from-scratch training benchmark, and transfer learning from BERT-base on several NLP datasets. Our work focused on the differences between sparse and dense training dynamics and their effect on optimal training, providing additional analysis towards the difficulty of sparse training. In our work we showed that very high levels of both sparsity and accuracy are possible simply by carefully adapting the number of training epochs and using sensible values for basic hyperparameters. We hope that these new results will encourage additional research on adapting training schedules, hyperparameters, optimizers, and data selection that will allow for the creation of sparsely-trained models that match these accuracy targets within a smaller training budget. We leave this as a challenge to the community.

References

Kale ab Tessera, Sara Hooker, and Benjamin Rosman. Keep the gradients flowing: Using gradient flow to study sparse network optimization, 2021.

Kyriakos Axiotis and Maxim Sviridenko. Sparse convex optimization via adaptively regularized hard thresholding. In *International Conference on Machine Learning (ICML)*, pp. 452–462. PMLR, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2019–2026, 2014. doi:10.1109/CVPR.2014.259.

Mathieu Blondel, Vivien Seguy, and Antoine Rolet. Smooth and sparse optimal transport. In *International conference on artificial intelligence and statistics*, pp. 880–889. PMLR, 2018.

Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5-6):629–654, 2008.

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

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semantic textual similarity-multilingual and cross-lingual focused evaluation. In *Proceedings of the 2017 SEMVAL International Workshop on Semantic Evaluation (2017). https://doi. org/10.18653/v1/s17-2001*, 2017.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

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. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021b.

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

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability, 2022.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper_files/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. *arXiv preprint arXiv:1906.10732*, 2019.

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.

Utku Evci, Max Vladymyrov, Thomas Unterthiner, Bart van Merrienboer, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information. In *International Conference on Learning Representations (ICLR)*, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

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

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

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

Noah Golmant, Zhewei Yao, Amir Gholami, Michael Mahoney, and Joseph Gonzalez. pytorch-hessian-eigenthings: efficient pytorch hessian eigendecomposition, 2018. URL `https://github.com/noahgolmant/pytorch-hessian-eigenthings`.

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

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

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.

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

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

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.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Shaoyi Huang, Dongkuan Xu, Ian EH Yen, Yijue Wang, Sung-En Chang, Bingbing Li, Shiyang Chen, Mimi Xie, Sanguthevar Rajasekaran, Hang Liu, et al. Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm. *arXiv preprint arXiv:2110.08190*, 2021.

Eugenia Iofinova, Alexandra Peste, and Dan Alistarh. How well do sparse imagenet models transfer? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Eugenia Iofinova, Alexandra Peste, and Dan Alistarh. Bias in pruned vision models: In-depth analysis and countermeasures. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Ajay Jaiswal, Haoyu Ma, Tianlong Chen, Ying Ding, and Zhangyang Wang. Training your sparse neural network better with any mask. In *International Conference on Machine Learning (ICML)*, 2022.

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

Siddhant M Jayakumar, Razvan Pascanu, Jack W Rae, Simon Osindero, and Erich Elsen. Top-KAST: Top-K always sparse training. *arXiv preprint arXiv:2106.03517*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better ImageNet models transfer better? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Eldar Kurtic and Dan Alistarh. Gmp*: Well-tuned global magnitude pruning can outperform most bert-pruning methods. *arXiv preprint arXiv:2210.06384*, 2022.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, 2022.

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.

Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. ffcv. `https://github.com/libffcv/ffcv/`, 2022.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. SNIP: Single-shot network pruning based on connection sensitivity. *International Conference on Learning Representations (ICLR)*, 2019.

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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

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.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning (ICML)*, 2021.

Tianlin Liu, Joan Puigcerver, and Mathieu Blondel. Sparsity-constrained optimal transport. *arXiv preprint arXiv:2209.15466*, 2022.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Yucheng Lu, Shivani Agrawal, Suvinay Subramanian, Oleg Rybakov, Christopher De Sa, and Amir Yazdanbakhsh. Step: Learning n:m structured sparsity masks from scratch with precondition, 2023.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint ArXiv:1306.5151*, 2013.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning (ICML)*, pp. 2498–2507. PMLR, 2017.

NeuralMagic. The DeepSparse Inference Engine. https://github.com/neuralmagic/deepsparse, 2022.

Mahdi Nikdan, Tomasso Pegolotti, Eugenia Iofinova, Eldar Kurtic, and Dan Alistarh. Sparseprop: Efficient sparse propagation for faster training of neural networks. *arXiv preprint arXiv:2302.04852*, 2023.

Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pp. 1447–1454, 2006.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. AC/DC: Alternating compressed/decompressed training of deep neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020.

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*, 115(3):211–252, 2015.

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.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020a.

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

Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. *arXiv preprint arXiv:1912.04427*, 2021.

Jonathan Schwarz, Siddhant Jayakumar, Razvan Pascanu, Peter Latham, and Yee Teh. Powerpropagation: A sparsity inducing weight reparameterisation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient second-order approximation for neural network compression. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Antoine Vanderschueren and Christophe De Vleeschouwer. Are straight-through gradients and soft-thresholding all you need for sparse training? In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Han Vanholder. Efficient inference with TensorRT. NVIDIA GTC On-Demand. Slides available at https://on-demand-gtc.gputechconf.com/gtcnew/sessionview.php?sessionName=23425-efficient+inference+with+tensorrt, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Richard von Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung . *Zammzeitschrift Fur Angewandte Mathematik Und Mechanik*, 1929.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Lu Yin, Gen Li, Meng Fang, Li Shen, Tianjin Huang, Zhangyang Wang, Vlado Menkovski, Xiaolong Ma, Mykola Pechenizkiy, and Shiwei Liu. Dynamic sparsity is channel-level sparsity learner. *arXiv preprint arXiv:2305.19454*, 2023.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. Prune once for all: Sparse pre-trained language models. *arXiv preprint arXiv:2111.05754*, 2021.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *International Conference on Machine Learning*, pp. 26809–26823. PMLR, 2022.

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.

Part I

Appendix

Table of Contents

A Training hyperparameters

We adopt and scale the training recipes proposed in the FFCV library (Leclerc et al., 2022). All runs use a linear decaying learning rate with a peak value $\eta_{\max} = 0.5$ at 2nd epoch that is gradually decreased to 0 at the end of the training. We train the models with batch size of 1024.

We use progressive resizing with training on random resized crops (RRC) of size 160×160 for the first 80% of training and the remaining 20% of training with crops of size 192×192. We run classifier on 224×224 center crops for the image resized to 256 px on validation following the standard ImageNet evaluation protocol. Differently from the FFCV recipes, we do not use test-time augmentation with averaging of the prediction on the original image and its flipped version since it requires 2x more inference FLOPs. For MobileNet experiments we adopted smaller value of weight decay 3e-5 following the (Kusupati et al., 2020) work and learning rate $\eta = 1.024$.

Table A.1: Augmentation and regularization procedure used in the work.

Method	Value
Weight decay	1e-4
Label smoothing ε	0.1
Dropout	✗
H.flip	✓
RRC	✓
Blurpool	✓
Progressive resizing	✓
Test crop ratio	0.875

B Sparse training algorithm hyperparameters

All sparse training methods use the same optimizer and learning rate scheduler hyperparameters and differ only in the specifics of weight pruning / regrowth procedure. In our work we prune only weights of convolutional (we do not prune biases and batch norm parameters) and keep the first convolution as well as the classification head (the last linear layer outputting the logits for classes) dense. This setup is common in the literature.

AC/DC In our AC/DC training setup we train the model without sparsity for the first 10% of the training and then alternate between sparse and dense training each 5 steps following the original paper (Peste et al., 2021). We run the last compression step and last decompression step for 10 and 15 epochs respectively, again following the prescription from the paper. We have ablated the duration of the compression and decompression phases and 5 epochs appeared to yield the best performance.

RigL In the original paper (Evci et al., 2020) authors train the models with fixed target sparsity and periodically update the fraction of connections in each pruned layer l following the cosine decay rule:

$$\frac{\alpha}{2}\left(1+\cos\left(\frac{\pi t}{T}\right)\right)(1-s_l) \tag{3}$$

Above s_l is the sparsity of a given layer l and α is the initial fraction of updated connections. The choice of update frequency ΔT has a strong impact on the method performance and in the original work authors obtained the best performance with $\Delta T \simeq 100-300$ steps, which corresponds roughly to 0.4-1.2 epochs on ImageNet with the batch size used in the work. In our work we updated connections every epoch to have setup close to the original work. We set $\alpha = 0.3$ as in the one used in the RigL work. Following the original work, we train with a fixed sparsity mask for the last 25% of training. For global sparsity we apply the ERK (Erdős–Rényi-Kernel) sparsity profile as in the original paper as it was shown to produce the best performance for a given sparsity.

GMP We gradually increase the sparsity from 0% (dense model) to the target sparsity following a cubic interpolation law. Sparsity is increased every 5 epochs. We train with a fixed mask for the last quarter of training duration as in RigL.

C Extended undertraining analysis

In Section 4.1, we examined the connection between sparsity, training loss, and validation accuracy in ResNet-50 models trained on Imagenet. Here, we present an alternative metric to the training loss, in the form of Entropy on the *validation* set, as well as validate our results on the CelebA dataset.

C.1 The validation entropy metric

Low prediction entropy implies that the prediction weight is largely concentrated in a single class, while a high entropy suggests that it is spread out over several classes. Intuitively, the entropy of the model is related to its "confidence" in predictions, and is independent of whether the predictions are correct (and so can be measured on data for which labels are not available). Conversely, low training loss measures the model's fit to the training data. The prediction entropy on test or validation data is also closely connected to the model's calibration.

The similarity of the two metrics can be seen directly from their formulas. We compute the cross-entropy loss and prediction entropy by taking the softmax over the vector of output values of the network and then applying the respective standard formulas, where the cross-entropy is taken with respect to the correct label distribution for the model (1 for the correct class and 0 otherwise). For an output of a network outputting a vector $Z = (z_1, z_2, ..., z_C)$ of size C with correct label L, the entropy H and the cross-entropy CE are given by the following formulas:

$$H(Z) = -\sum_{i=1}^{C} \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \log \left(\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \right) \quad \text{and} \quad CE(Z) = -\log \left(\frac{e^{z_L}}{\sum_{j=1}^{C} e^{z_j}} \right). \tag{4}$$

We expect a sufficiently large and well-trained model to have (a) low loss on the training data and (b) fairly low average prediction entropy, while a model that is not well-trained to have high prediction entropy. However, as is conventionally known, continued training on dense and low-sparsity models resulting in overfitting will lower these metrics further.

The experimental setup is exactly as in Section 4.1, just with the additional computation of the Entropy metric on the validation set. We present the results in Figure C.1, which also includes the plots from Figure 1 for ease of comparison.

We observe that, for all sparsity levels, the Train Loss and Validation Entropy metrics behave very similarly. Most crucially, we observe that, just as with the training loss, there is an 'optimal' entropy level across model sparsities, with models whose validation entropies fall below that value having lower validation accuracy. We believe this provides additional evidence that these metrics may be used to detect overfitting in models and to find the optimal training duration.

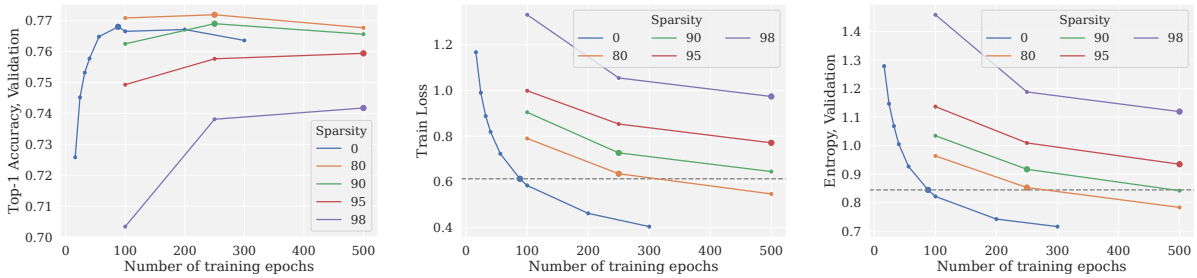

Figure C.1: Average validation accuracy (left), Train loss at final epoch (center), and Entropy (right) for sparse and dense ImageNet models trained for different numbers of epochs. The highest-accuracy model for each sparsity level is highlighted with a larger marker. The cross-entropy loss and entropy level of the dense model is also shown with a dashed line, to simplify comparison.

C.2 CelebA results

To validate our findings, we repeat this experiment on the Celeb-A Dataset (Liu et al., 2015). This dataset consists of a combined 202599 face images of 10177 celebrities collected from the public domain, automatically

cropped to the face, and annotated with 40 binary labels. Due to its content, this dataset is frequently used to study bias in machine learning models, and has also been used in studies on the effect of sparsity on bias (Hooker et al., 2019; Iofinova et al., 2023).

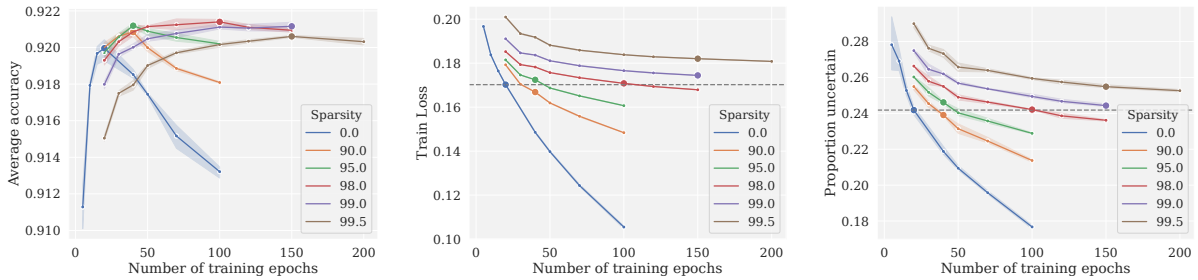

Figure C.2: Average validation accuracy (left), train loss at final epoch (center), and uncertainty (right) for sparse and dense CelebA models trained for different numbers of epochs. The highest-accuracy model for each sparsity is highlighted with a larger marker. The cross-entropy loss and entropy level of the dense model is also shown with a dashed line, to simplify comparison.

Experimental Setup. Following (Iofinova et al., 2023), we train the ResNet18 architecture on this task. We train dense and sparse (90%, 95%, 98%, 99%, and 99.5%) models for a varying number of epochs (5-200); in some cases the very low or very high-epoch runs are skipped if it is clear that the duration will not be optimal for that sparsity. Sparse models are produced with a variant of AC/DC, in which the sparsity of the sparse phases ramps up progressively from 90% to the final target sparsity; this is necessary to prevent layer collapse at very high sparsities. Unlike the ImageNet experiments, here the phase length varies somewhat with duration, due to the extremely short duration of some runs. For each experiment, characterized by a sparsity/epoch-length pair, we measure the accuracy and training loss of the resulting model. In addition, following (Iofinova et al., 2023), we measure the *uncertainty*, rather than entropy, of the model predictions on the test set. The prediction uncertainty is computed as follows: first, the sigmoid operator is applied to each logit's output in order to obtain a pseudo-probability of a positive label between 0 and 1; if this quantity is between 0.1 and 0.9, the prediction is considered uncertain. We then compute the proportion of uncertain predictions across the validation dataset.

Results. The results are presented in Figure C.2. We observe that, consistent with our earlier observations on ImageNet, the optimal training duration goes up with sparsity, with dense models reaching their optimal accuracy at 25 epochs, and 99% and 95% sparse models at 150 epochs. Further, we observe that, even as training loss and test uncertainty always decrease with longer training, the overall training loss and proportion of uncertain predictions goes up with sparsity at a fixed training length. As in the ImageNet example, the highest-performing models at each sparsity have a similar training loss of about 0.17 and mean prediction uncertainty of about 24%, except for the very sparse 99.5% model, which has slightly higher 26% uncertain predictions.

Discussion. We interpret these results as corroborating evidence that sparse models require longer training compared to dense models to achieve optimal accuracy before the overfitting starts to take place.

D MobileNet results

In addition, we conducted sparsification of the MobileNet-V1 model (Howard et al., 2017), a CNN optimized for inference on mobile devices. We applied AC/DC for 1000 epochs with sparsity targets 75% and 90% using a similar training recipe to ResNet50 except for some differences specified in Appendix A. To achieve the best results we do not prune the input convolution, as well as the classification head and depthwise convolutions, due to their minor contribution to the overall amount of FLOPs and significant impact on the performance of the model.

Method	Top-1 accuracy (%)		Relative Drop	Sparsity
	Dense (D)	Pruned (P)	$100 \times \frac{(P-D)}{D}$	
AC/DC++	72.74	72.49	-0.25	75.00
	72.74	70.80	-1.94	90.00

Table D.2: Sparse training of MobileNet-V1 with AC/DC++ on ImageNet-1k.

With a longer training recipe one can achieve almost negligible accuracy drop at 75% sparsity and moderate performance decrease at 90%. The results can be found in Table D.2.

E Impact of extended training on transfer performance

We further validate our results by using sparse and dense ResNet50 models pretrained on ImageNet for transfer learning for other vision recognition tasks. Transfer learning is a common paradigm in which a large dataset is used to set network weights before they are further fine-tuned on the (typically, smaller) dataset of interest; this approach can provide significant gains over direct training on the smaller task from a random initialization. In this section, we refer to the larger dataset/task (in our case, ImageNet) as the *upstream* task, and the smaller dataset/task as the *downstream* task.

There are two common approaches to transfer learning, the choice of which largely depends on the technological capabilities of the system doing the learning task. If a large amount of compute is available, all model weights can be trained on the downstream task, in a process we call *full finetuning*. Otherwise, if compute is limited, all the layers of the deep neural network except for the final classifier are frozen after downstream training and used purely as a feature extractor, and only the final layer (properly resized) is trained on the downstream task, as a linear classifier on the extracted features. We call this process *linear finetuning*.

We apply both transfer learning approaches to a standard set of twelve downstream tasks, which are frequently used as benchmarks for transfer learning performance(Kornblith et al., 2019; Iofinova et al., 2022; Salman et al., 2020). The twelve datasets include six specialized tasks and six general tasks; a full list of downstream tasks is given in Appendix Table E.4.

We use four variants of ResNet50 models pretrained on ImageNet as the upstream model: dense and 95% globally sparse models trained for 100 and 1000 epochs. Otherwise, we follow the training hyperparameters of (Iofinova et al., 2022), and, also following this work, we compute a single metric across all twelve tasks by computing Average Increase in Error for a set of tasks T over a baseline model. This metric is computed as follows: for each of the downstream tasks, we compare the difference in the error (1−Top-1 accuracy) when the baseline (100-epoch dense) model is used for transfer learning, versus when another model (either sparse, or extended-training dense) is used. The increase in error is the difference of these two quantities divided by the dense error; the final metric is the average of these relative differences across all downstream datasets. As argued in (Iofinova et al., 2022), normalizing error differences by the dense error allows us to roughly equalize dataset impact when we compute a single metric across different datasets with very different model performance (e.g., nearly 100% accuracy on the Pets dataset, but about 50% accuracy on the Aircraft dataset. Formally, the metric is computed as:

$$AIE = \frac{1}{|T|} \sum_{t \in T} \frac{Err_{Model,t} - Err_{Baseline,t}}{Err_{Baseline,t}} \tag{5}$$

We compute the AIE using the 100-epoch dense model as the baseline model, and present the results in Table E.3. We observe that the effect of extended training on the dense model is neutral at best - there is a small increase in error with extended training in the linear finetuning regime, and no change in the full finetuning regime. Conversely, the 1000 epoch 95% sparse model outperforms the 100-epoch one across both regimes, with a higher error drop in the linear finetuning regime and a smaller error increase in the full finetuning regime.

Training Duration	Linear - Dense	Linear - 95% Sparse	Full - Dense	Full - 95% Sparse
100ep.	-	-0.123	-	0.112
1000ep.	0.026	-0.182	0.000	0.062

Table E.3: Transfer learning for image recognition - Average Increase in Error over the 100-epoch dense model. Lower is better: positive numbers indicate that the model has, on average, more error than the baseline model; negative numbers indicate that the model has, on average, less error.

Dataset	Number of Classes	Train/Test Examples	Accuracy Metric
SUN397(Xiao et al., 2010)	397	19 850 / 19 850	Top-1
FGVC Aircraft(Maji et al., 2013)	100	6 667 / 3 333	Mean Per-Class
Birdsnap(Berg et al., 2014)	500	32 677 / 8 171	Top-1
Caltech-101(Li et al., 2004)	101	3 030 / 5 647	Mean Per-Class
Caltech-256(Griffin et al., 2006)	257	15 420 / 15 187	Mean Per-Class
Stanford Cars(Krause et al., 2013)	196	8 144 / 8 041	Top-1
CIFAR-10(Krizhevsky et al., 2009)	10	50 000 / 10 000	Top-1
CIFAR-100(Krizhevsky et al., 2009)	100	50 000 / 10 000	Top-1
Describable Textures (DTD)(Cimpoi et al., 2014)	47	3 760 / 1 880	Top-1
Oxford 102 Flowers(Nilsback & Zisserman, 2006)	102	2 040 / 6 149	Mean Per-Class
Food-101(Bossard et al., 2014)	101	75 750 / 25 250	Top-1
Oxford-IIIT Pets(Parkhi et al., 2012)	37	3 680 / 3 669	Mean Per-Class

Table E.4: Datasets used as downstream tasks for transfer learning with computer- vision models.

F Evaluation on ImageNet-C

In order to confirm the robustness of our models against perturbations, we evaluate AC/DC and dense models on the ImageNet-C(Hendrycks & Dietterich, 2019) dataset, which consists of the standard ImageNet validation dataset to which perturbations such as noise, blur, photographic effects (such as contrast enhancement) or weather conditions (such as snow or fog) were digitally added. We use the lowest "1" level of perturbation, as we find that the quality already drops considerably over the clean data. Our results are shown in Figure F.4. We observe that, as expected and also as reported in (Liebenwein et al., 2021), dense models outperform sparse ones on corrupted data. Further, and consistently with our other findings, extended training on the dense model does not improve performance on the ImageNet-C dataset; conversely, such improvement can be seen for all 19 perturbation categories for 95% sparse models.

G Comparison with the small dense model

One may ask whether one can achieve similar performance with a smaller dense model. We have taken a smaller version of ResNet50 with 2x smaller hidden dimension (denoted ResNet-50x0.5). We trained the model with the same training procedure as the baseline ResNet-50 model for 1000 epochs. One can observe from Table G.5 that even for smaller number of FLOPs sparse model is much better than the dense one.

Table G.5: Small-dense vs sparse model on ImageNet-1k

Model (%)	Sparsity (%)	FLOPs	Accuracy (%)
ResNet-50x0.5	0	-	69.8
ResNet-50	95	-	77.5

H AC/DC dense fraction ablation

The original AC/DC paper has equal length of compression and decompression phases and one may raise a question, whether it would be better to train the model longer in compressed state or decompressed to

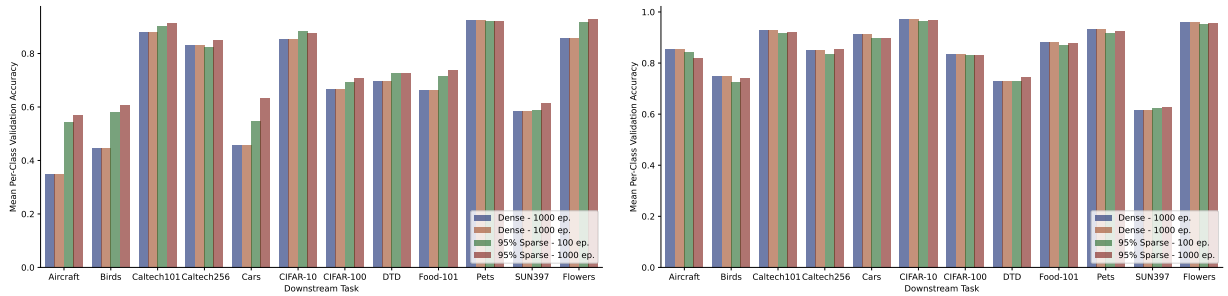

Figure E.3: Linear (left) and Full-network (right) transfer learning mean per-class accuracy for individual datasets.

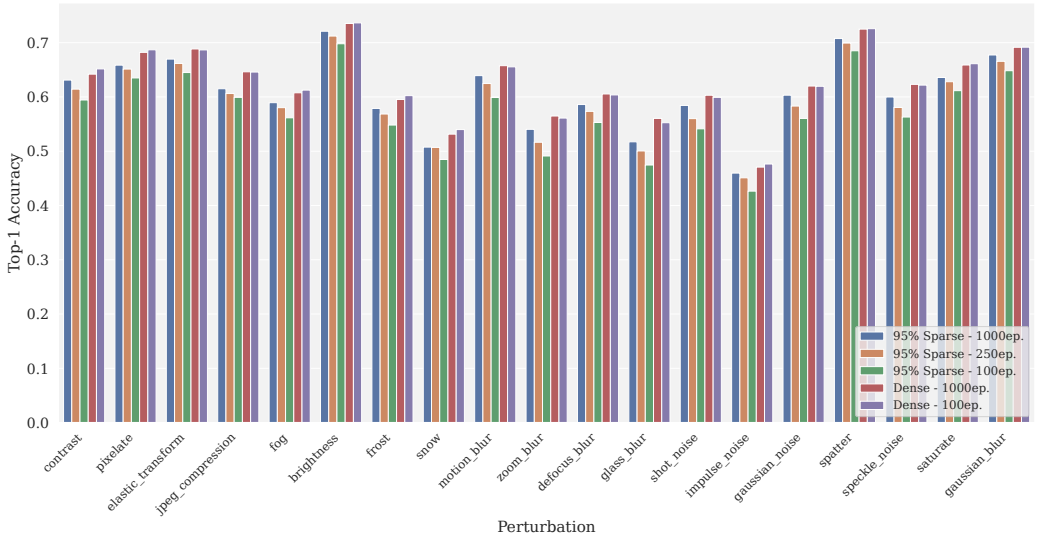

Figure F.4: Linear (left) and Full-network (right) transfer learning mean per-class accuracy for individual datasets.

achieve better performance for a given sparsity target. In the experiments below we train the ResNet-50 model for 1000 epochs with 95% sparsity. It turns out that allocating equal time for dense and sparse training works the best, as one can see from the Table H.6. Here 0 means pruning at some moment of training and finetuning with fixed mask and 1 is the accuracy of dense model. Even when training most of the time in sparse regime one still doesn't lose much in performance.

Table H.6: Ablation on the sparse training fraction.

Sparse training fraction (%)	Accuracy (%)
0	79.0
0.2	77.2
0.5	77.5
0.8	77.3
1	75.0

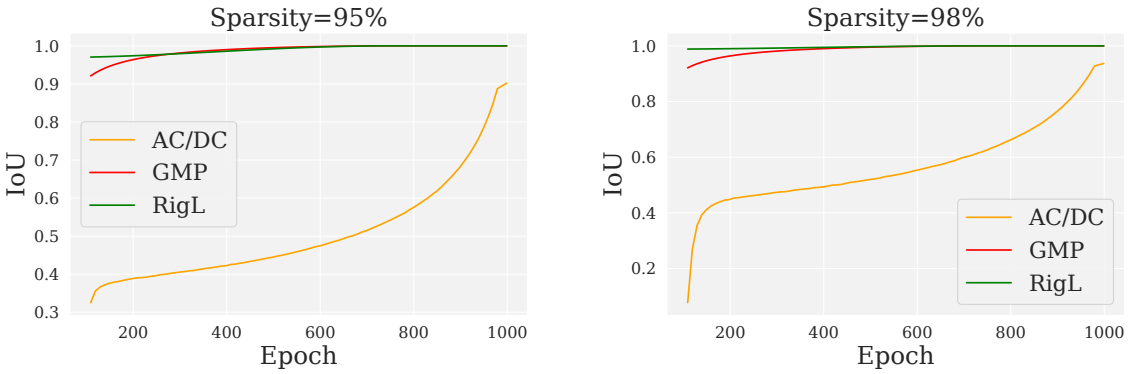

Figure J.5: Mask IoU between two consecutive checkpoints.

I AC/DC phase duration ablation

Another hyperparameter that may have a significant impact on performance is the update interval of AC/DC iteration. Intuitively, more frequent updates may allow for more changes in the choice of sparsity of the pruning mask and at the same longer training with fixed mask can be needed to the weights to adapt well for a given sparsity pattern. We have tried different choices of AC/DC iteration duration for ResNet-50 trained for 1000 epochs and 95% global sparsity and the chosen frequency update of 5 epochs is close to the optimum.

Table I.7: AC/DC iteration length ablation.

Phase length duration (epochs)	Accuracy (%)
1	77.0
5	77.5
10	77.2
50	75.9

J Mask analysis

Sparse methods considered in our work differ in the amount of sparsity mask exploration. GMP gradually increases sparsity; once the weight is pruned, it is never reintroduced. RigL decreases the fraction of updated parameters following the cosine annealing rule:

$$f_{decay}(t;\alpha;T_{end}) = \frac{\alpha}{2}\left(1+\cos\left(\frac{\pi t}{T_{end}}\right)\right) \tag{6}$$

This fraction of connections is dropped and reintroduced in a single step. AC/DC makes all parameters trainable on decompression phases, therefore any parameter could be potentially reintroduced. However, as shown later, some fraction of weights remains zero even on decompression. To measure the difference between two consecutive sparsity masks we compute their IoU (Intersection over Union), the amount of parameters that are nonzero for both checkpoints divided by the amount of parameters that are nonzero in either of checkpoints. High IoU value (close to 1) means that two sparsity masks overlap significantly, whereas low IoU (close to 0) implies low similarly between sparsity masks.

We have taken checkpoints saved on the $109^{th}, 119^{th}, \ldots 999^{th}$ epoch (taken at the end of every AC/DC step and at the same epochs for other methods for a consistent comparison) collected during 1000 epochs runs with 95% and 98% target sparsity and measured IoU between two consecutive masks for parameters being pruned (skipping biases and batch norm parameters). For GMP and RigL, mask IoU can be computed analytically based on the update rule. The evolution of sparsity mask IoU during training is presented on

Figure J.5. One can observe that AC/DC shows significantly stronger mask exploration compared to GMP and RigL. This behavior could account for the better performance of AC/DC as a sparse trainer.

K Structured sparsity in unstructured sparse models

		Channel sparsity			
Sparsity	Epochs	`layer1.0.conv2`	`layer2.1.conv2`	`layer4.2.conv2`	avg
80	100	0	0	0	1.07
	1000	18.75	0	35.35	4.91
90	100	4.69	1.56	0.78	3.58
	1000	31.25	19.53	61.91	10.68
95	100	0	7.03	22.85	8.8
	1000	26.56	22.66	80.27	16.76
98	100	7.81	12.5	75.59	23.65
	1000	48.44	37.5	96.68	27.93

Table K.8: Fraction of zero output channels for specific layers and on average.

We mainly consider only *unstructured* sparsity, therefore groups of weights and entire channels in particular do not have to be sparse. However, we observed that some of the channels in convolutional kernels are entirely pruned. The effect becomes more pronounced with higher sparsity and longer training.

In Table K.8, we show the fraction of sparse *output* channels for layers in the front, middle and the end of the model and the global fraction of zero channels across all convolutional layers in the model. We observe that channel sparsity increases proportionally with the unstructured sparsity target and with training time, and that, for high sparsity, we obtain a very high proportion of zeroed-out output channels, especially in the wider bottom layers. This is in tune with previous work observing the emergence of structured sparsity in dynamic sparse training (Peste et al., 2021; Iofinova et al., 2022; Yin et al., 2023). We provide a first explanation for this behavior in the next section.

L Impact of weight decay on sparsity and model performance

Recall that AC/DC makes all parameters trainable on decompression, therefore, one might expect that sparsity on decompression phase would be near zero. However, we observed that a large fraction of weights remains zero even when the sparsity mask is not imposed. This effect is linked to the channel sparsity discussed in the previous section: once a channel is completely zeroed out, it will continue to receive zero gradients even when the sparsity mask is removed. Further, we provide evidence that this phenomenon is linked to increasing the weight decay mechanism, and in particular the value of this parameter: intuitively, weight decay slowly drives weights towards zero; whenever a full channel is zeroed out in the compression phase, it remains "captured" under the sparsity mask until the end of training.

We investigated this empirically by training ImageNet models on ResNet50 with 95% compression sparsity using AC/DC++ for 100 epochs, varying the weight decay parameter from 10^{-6} to 10^{-3}. Our results are shown in Figure L.6. We observe that the fraction of zero parameters increases with the magnitude of weight decay and also over the course of training. Concretely, we observe that all weight decay values lead to almost fully dense models during the first decompression phase. From there, very low weight decay values of 10^{-6} and 10^{-5} lead to very little sparsity during the next two decompression phases, and about 10% sparsity during the final five. Conversely, very high weight decay of 10^{-3} leads to an immediate jump to 50% sparsity during the second decompression phases, which then increases to 60% over the rest of training. The intermediate value of 10^{-4}, which is the standard setting and was used in our experiments, leads to an intermediate sparsity, which gradually rises to about 24% over successive decompression phases.

We further present the accuracy of the resulting models in Table L.9. We observe that properly setting the weight decay hyperparameter is crucial for good performance of AC/DC++, and confirm that the standard value 10^{-4} is close to the optimal value, as the Table L.9 shows.

Sparse Decompression. Building on this observation, we ask how imposing a fixed minimal sparsity also on *decompression stage* impacts the final performance. We conducted a few 100-epoch long AC/DC experiments with 95% target sparsity and set the sparsity on decompression to some fixed value, smaller than compression sparsity. We observe that the performance is almost unaffected up to 80% decompression sparsity, showing that full mask exploration is not necessary during training.

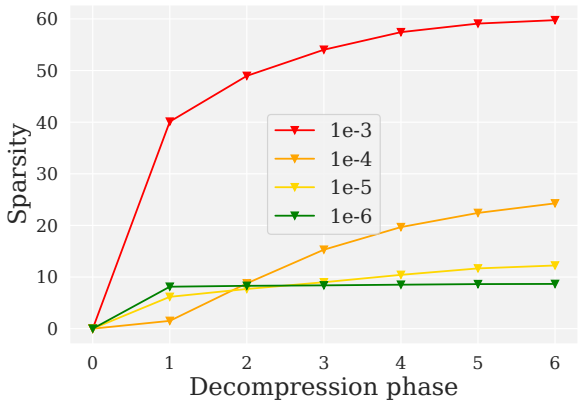

Figure L.6: Sparsity on decompression phases for 100-epoch AC/DC++ runs with varying values of weight decay. We point out that on decompression phases no sparsity is enforced.

Weight decay	Top-1 accuracy (%)
10^{-6}	70.52
10^{-5}	73.54
10^{-4}	74.90
10^{-3}	68.57

Table L.9: Accuracy vs weight decay.

Decompression sparsity	Top-1 accuracy (%)
0	74.82
50	75.09
60	74.81
70	74.83
80	74.38

Table L.10: Accuracy vs decompression sparsity.

M Loss landscape analysis

In order to get more insights into the reasons for the difficulty of optimizing sparse neural networks, we investigate two properties of the loss landscape. First, we measured *landscape sharpness at the end of the training*, defined as the maximal eigenvalue of the Hessian matrix, for all sparse training methods considered, various sparsities and number of training epochs and compared with the one of standard dense training. Second, we interpolated the training and validation loss between checkpoints obtained at intermediate steps throughout the 1000 epoch AC/DC run with 95% target sparsity.

The largest Hessian eigenvalue was estimated via the power iteration method based on Hessian-vector products using a customized version of the Eigenthings library (Golmant et al., 2018). To compute the largest Hessian eigenvalue we ran power iteration method (von Mises & Pollaczek-Geiringer, 1929) for 20 iterations using the modified version of the code from (Golmant et al., 2018) package. 20 iterations were usually enough for the iterations to converge. Hessian is computed only with respect to the non-zero weights. Technically, for a current estimate of a leading eigenvector $\mathbf{e}_i$ and sparsity mask $\mathbf{m}$ we compute $\nabla(\mathbf{m} \cdot \nabla(\mathbf{m} \cdot \mathbf{e}_i)) = \mathbf{H}_\mathbf{m}$ which is equivalent to the hessian-vector product for a truly sparse model. In Figure **??**, we observe that, across all methods, sharpness increases with the length of the training run, indicating that sharper minima require extended training to be reached via SGD. Additionally, sharpness decreases with the increase of sparsity. All sparse training methods attain lower sharpness compared to the dense model. Models trained with AC/DC and RigL have slightly lower sharpness compared to models trained with GMP, presumably because the former two methods manage to reach flatter optima which are conjectured to have better generalization properties (Merity et al., 2017).

To examine mode connectivity behavior, in Figure M.7, we connected the checkpoint obtained on the $99^{th}, 199^{th}, \ldots 999^{th}$ epoch via piecewise-linear curves with 10 points between checkpoints. A notable observation is that all checkpoints are separated by a loss barrier, whose height is increasing with the number of epochs. Probably, this behavior is a manifestation of the progressive sharpening phenomenon (Cohen et al., 2022) where the model sharpness is increasing gradually with training until reaching the peak value and then

plateaus. Yet, for sparse models, the duration of the longest runs is not long enough to reach the sharpness plateau.

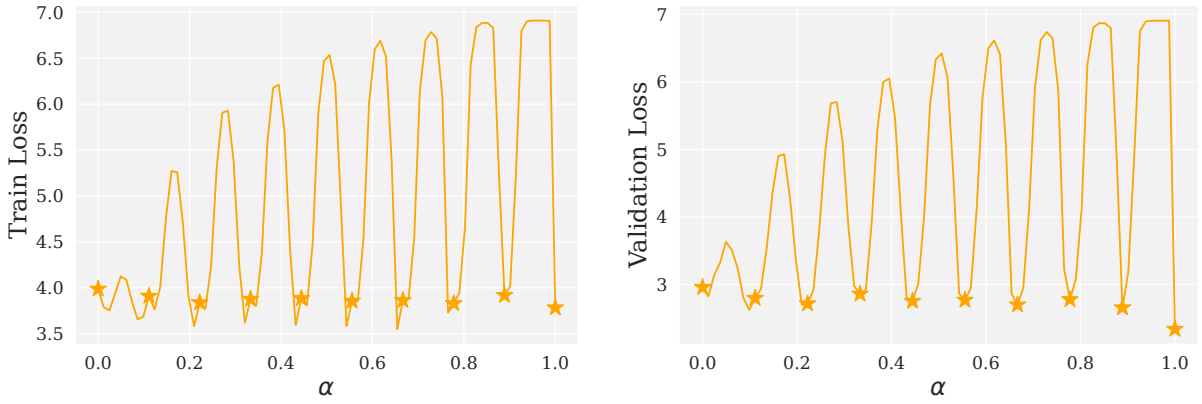

Figure M.7: Loss interpolation curves on the training (**left**) and validation (**right**) checkpoints of the 95% sparse 1000-epoch AC/DC model. α corresponds to the fraction of paths traversed from the first checkpoint to the last. Stars denote checkpoints between which loss is interpolated.

N Different sparsity patterns.

Since the ResNet models increase the number of channels with the decrease of feature map resolution, one would expect that bottom layers (those with more channels) should be pruned more aggressively compared to the one with less channels. Here our results are consistent with the known observations from literature that global sparsity achieves higher performance for the same sparsity. In addition, we have carried out experiments with the more hardware-friendly block-4 sparsity pattern.

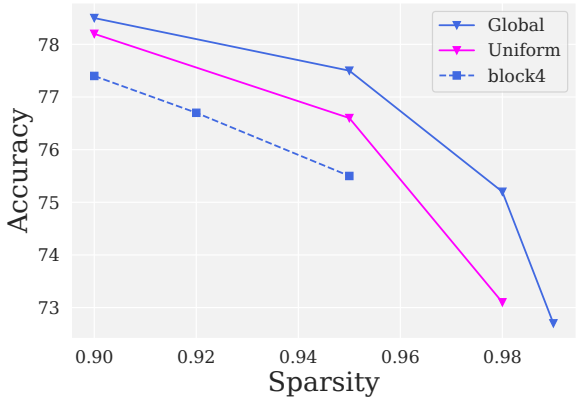

Figure N.8: Accuracy vs sparsity for different sparsity distributions. Block4 denotes global pruning with weights pruned in groups of 4.

O Transfer learning datasets for language models

In Table O.11 we provide a brief summary of datasets in the popular GLUE benchmark (Wang et al., 2018). Following previous work (Sanh et al., 2020a; Zafrir et al., 2021; Kurtic & Alistarh, 2022; Kurtic et al., 2022; Zhang et al., 2022; Huang et al., 2021), we exclude WNLI dataset from consideration in all experiments.

Dataset	Train/Test Examples	Accuracy Metric
RTE (Wang et al., 2018)	2.5k / 3k	Accuracy
QNLI (Wang et al., 2018)	105k / 5.4k	Accuracy
MRPC (Dolan & Brockett, 2005)	3.7k / 1.7k	F1 score and Accuracy
SST-2 (Socher et al., 2013)	67k / 1.8k	Accuracy
CoLA (Warstadt et al., 2019)	8.5k / 1k	Matthews correlation coefficient
STS-B (Cer et al., 2017)	7k / 1.4k	Pearson and Spearman correlation
MNLI (Williams et al., 2017)	393k / 20k	Matched (m) and mismatched (mm) accuracy
QQP (Wang et al., 2018)	364k / 391k	Accuracy and F1 score

Table O.11: Datasets used as downstream tasks for transfer learning with language models.

P Additional plots

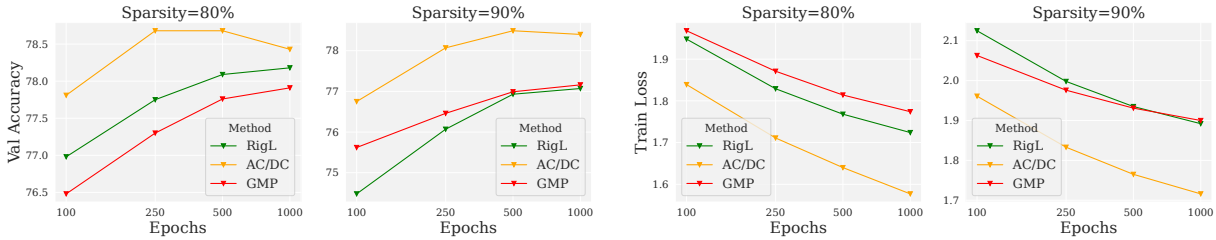

Figure P.9: (**left**) Validation accuracy on ImageNet-1k vs number of epochs for different sparse training methods. (**right**) Training loss on ImageNet-1k vs number of epochs for different sparse training methods.