RETHINKING SPARSE SCALING THROUGH THE LENS OF AVERAGE ACTIVE PARAMETER COUNT

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter pruning has emerged as a promising technique to address the growing computational demand of large language models (LLMs). While many studies focus on post-training pruning of LLMs, sparse pre-training offers a compelling alternative: sparsifying during pre-training reduces both training and inference costs. In this work, we conduct the first comprehensive study on optimal sparse pre-training configurations for LLMs, exploring various pruning schedules across different sparsity levels and training duration. We evaluate 80 unique configurations and find that a pruning schedule starting at 25% of total training compute and ending at 75% achieves near-optimal final evaluation loss. Our findings provide valuable insights for efficient and effective sparse pre-training of LLMs. Furthermore, we propose a new scaling law that modifies the Chinchilla scaling law to use the *average* number of active parameters during training. We present both empirical and theoretical evidence that this modification accurately models evaluation loss for both sparsely and densely pre-trained LLMs, thus offering a unified scaling law for dense and sparse model training. Our insights suggest that, while sparse pre-training yields similar model loss as dense pre-training for the same compute budget, it offers a clear advantage: the final model is smaller, resulting in significant potential computational savings during inference.

1 INTRODUCTION

Research consistently shows that larger language models, trained on more data, achieve better performance (Brown et al., 2020; Kaplan et al., 2020; Hoffmann et al., 2022; Nakkiran et al., 2019). However, their enormous size poses an increasingly pressing challenge to their efficient deployment and equitable access. One promising approach to addressing these challenges is sparse training, which reduces the computational burden by using only a subset of the neural network parameters both during training and inference. This technique, closely related to neural network pruning (Han et al., 2015; LeCun et al., 1989; Hassibi et al., 1993; He et al., 2017), gained prominence with the Lottery Ticket Hypothesis (Frankle & Carbin, 2019), which provided compelling evidence for the feasibility of sparse training. Subsequent work has introduced efficient algorithms to realize the promised efficiency gains (Evci et al., 2021; Peste et al., 2021; Kuznedelev et al., 2024).

While a growing body of research investigates pruning pre-trained large language models (LLMs) (Sun et al., 2024; Frantar & Alistarh, 2023; Xia et al., 2024), our work focuses on sparsely pre-training LLMs. The significant computational and engineering cost associated with LLM pre-training itself, in addition to designing the pruning algorithm presents substantial obstacle to this line of research. For example, identifying sparse sub-networks that can be trained to good performance as suggested by the Lottery Ticket Hypothesis (Frankle & Carbin, 2019) typically involves iterative pruning and retraining, a process that becomes prohibitively expensive at the scale of large language models. This expense effectively limits investigation to smaller-scale models, leaving the optimal strategies for sparse pre-training of LLMs largely unknown. One way to bridge this gap and extend small-scale insights to the realm of large models is through the analysis of scaling laws.

This leads to a critical question: how does sparsity influence the scaling laws that govern large language model performance? Scaling laws have been instrumental in predicting the relationship

between the loss, model size, data size, and computational resources for dense models (Kaplan et al., 2020; Hoffmann et al., 2022). But how do these laws adapt to the introduction of sparsity?

Previous work on sparse pre-training by Frantar et al. (2023) introduced a new sparsity-aware law that departed from the established scaling laws for dense models. The modified laws included extra terms dependent on the final sparsity, aiming to capture the sparsity-specific scaling effects.

Our approach. Instead, we revisit the original dense scaling laws and explore how to set the parameter count term for sparse pre-training, during which we gradually remove model parameters. We show that the dense scaling law can effectively model sparse pre-training by updating the parameter count to reflect the varying number of active parameters. This suggests that the core principles of dense scaling laws remain applicable in the sparse pre-training regime, with the key adjustment being a more nuanced consideration of parameter count.

Optimal sparse pre-training configuration. To validate this approach, we evaluate over 80 combinations of sparse pre-training schedules, sparsity levels, and training durations. Our work provides the first systematic analysis of these sparse pre-training configurations across design dimensions critical to sparse pre-training. This comprehensive evaluation uncovers the optimal configurations and offers new insights into the dynamics of sparse training.

Our results show that, for a fixed compute budget, training the dense model for 25% of the total training compute and gradually removing weights over the next 50% leads to near-optimal final evaluation loss across various training duration and sparsity levels. We find that the optimal learning rates and batch sizes for sparse pre-training closely match those used for the original dense model under the same compute budget, where the original dense model is the starting model for sparse pre-training. We also provide additional analysis into failure modes that occur with non-optimal sparse pre-training configurations, highlighting the importance of a properly tuned configuration. Collectively, our analysis presents practical prescriptions to simplify the transition between dense and sparse training configurations.

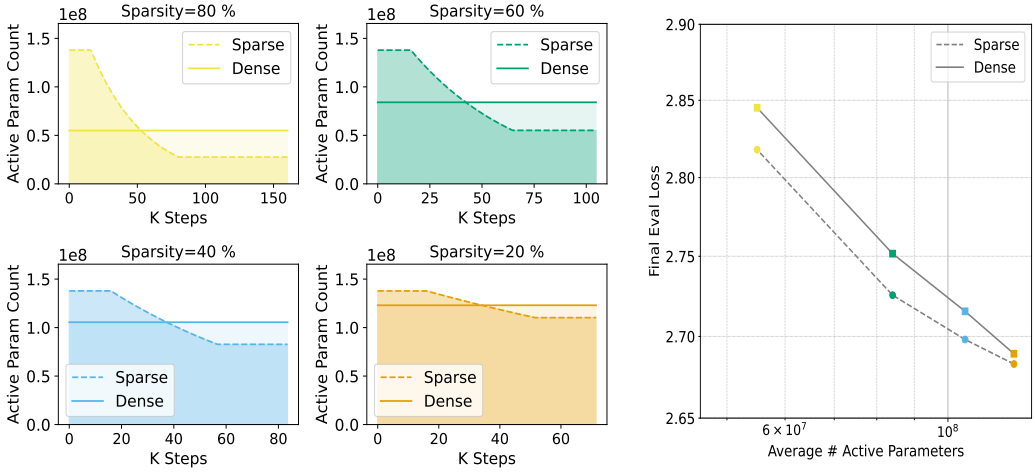


Figure 1: We show the predictive power of average active parameters by creating two families of models. The first is sparse, starting from a dense model with 138 million prunable parameters in the linear layers and targeting final sparsity levels of 20%, 40%, 60%, and 80%. The second is dense, created by adjusting the hidden dimension to match the average number of active parameters throughout sparse-pre-training for each sparse models. In the left plot, we represent sparse models with dashed lines and dense models with solid lines. Each sparse-dense pair, with matching average active parameters, is shown in the same subfigure. Each pairs of model shares the same total training compute. In the right plot, despite differences in pre-training techniques, sparse and dense models with matching average active parameters (indicated by matching colors) achieve similar final loss.

Scaling analysis. We propose a scaling law to model the final evaluation loss of sparse-pre-trained language models. Our scaling law builds on the Chinchilla scaling law by incorporating the *average* number of active parameters throughout pre-training, rather than the total number of dense parameters. A parameter is considered active if it receives gradient updates, whereas pruned parameters, which do not receive updates, are inactive. This adaptation allows us to predict evaluation loss across a range of model sizes, sparsity levels, and training regimes. Additionally, based on empirically verified assumptions about sparse pre-training dynamics, we provide a theoretical justification for using the average active parameters to model the evaluation loss of sparsely pre-trained language models.

Average active parameters. To demonstrate the predictive power of average active parameters for sparse scaling, we train four pairs of sparse and dense models. In each pair, we keep the average active parameters throughout pre-training [1] and the total training data the same. Despite the difference in active parameters during most of the pre-training, as illustrated in the left plot of Figure 1, each pair of the sparse and dense model achieve similar final loss, as shown in the right plot of Figure 1. These results show that, with all other factors held constant, the average number of active parameters is an effective predictor of the final evaluation loss for sparse models, just as it is for dense models.

Contributions. We make the following contributions in our work:

1. We bridge the gap between sparse and dense scaling laws by modifying the parameter count term in the Chinchilla scaling law, extending the same law to to model sparse pre-training. We show this modified scaling law accurately models the evaluation loss across a range of model sizes, sparsity levels and training duration.
2. We present a theoretical analysis, based on empirically validated assumptions about sparse pre-training dynamics, that justifies using the average number of active parameters to model the loss of sparsely pre-trained language models.
3. We search over 80 sparse pre-training configurations and present a simple prescription that achieves optimal or near-optimal loss across different sparsity levels and training duration.
4. We present an analysis over the failure modes when sparse pre-training configurations deviate from the said prescription, highlighting their practical importance.

Implications. Our work enables practitioners familiar with dense scaling laws to apply those same principles to sparse scaling. Together with prescriptions for optimal sparse pre-training configurations, our work aims to ease the transition from dense pre-training to sparse pre-training, promoting the development of more energy-efficient large language models.

2 Related Work

Neural scaling laws. Neural scaling laws provide a framework for understanding how neural networks' performance scales with parameters, data, and compute (Banko & Brill, 2001; Goodman, 2001; Ghorbani et al., 2021; Kaplan et al., 2020; Hoffmann et al., 2022; Bansal et al., 2022; Gordon et al., 2021). Kaplan et al. (2020) showed that for modern transformer-based language models, model loss decreases predictably with increasing model size, dataset size, and compute, following a power-law relationship. Hoffmann et al. (2022) later refined these insights by optimizing hyperparameter configurations, such as learning rate schedules, and proposed a new scaling law that emphasizes scaling training data more aggressively than Kaplan et al. (2020)'s original recommendations. Most relevant to our work are Rosenfeld et al. (2021) and Frantar et al. (2023). Rosenfeld et al. (2021) focused on small-scale CNNs for image classification, while Frantar et al. (2023) focused on transformer-based vision and language models; both modeled their respective performance as a function of model size and pruning configurations. Our work is different in that we unify the functional forms of scaling laws for both dense and sparse pre-training. This unification is partly enabled by our novel exploration of optimal hyperparameter configurations for sparse pre-training.

Additionally, our study is the largest-scale investigation of sparsely pretrained LLMs to date, with our largest model using over 5 times the compute of the largest model examined in prior work (Frantar et al., 2023). The largest model we investigate requires 4.5×10^{20} FLOPs training compute.

[1] We adjust the hidden dimension of dense models to match the sparse models, but since it must be divisible by the number of attention heads and chips, we create two dense models that approximate the target parameter count and linearly interpolate between them.

Pruning. Sparse pre-training involves pruning parameters in an LLM during the pre-training process. While there are many pruning algorithms available (LeCun et al., 1989; Hassibi et al., 1993; Han et al., 2015; He et al., 2017; Frankle & Carbin, 2019; Renda et al., 2020; Peste et al., 2021; Evci et al., 2021), we focus on a simple class of algorithms known as iterative magnitude pruning (Zhu & Gupta, 2017; Frankle & Carbin, 2019; Renda et al., 2020). Since sparse pre-training results in a sparse model at the end of training, our method can also be viewed as a pruning algorithm. In contrast to prevailing approaches that train a large dense model and then prune it while preserving accuracy (Frantar & Alistarh, 2023; Sun et al., 2024), our analysis shows that, within the sparsity levels we examined, it is possible to directly train a sparse model that achieves the same final loss using the same compute budget as training the large dense model. This simplifies the model development process by eliminating the need for pruning as a post-training step.

Dynamic parameter schedule. While practitioners typically pre-train LLMs with a fixed number of active parameters throughout the training process (Groeneveld et al., 2024; Shoeybi et al., 2020), a growing line of work explores varying this number during pre-training to improve computational efficiency (Yao et al., 2023; Panigrahi et al., 2024; Yano et al., 2024). This line of work often focuses on gradually increasing the number of parameters during training. Yao et al. (2023) proposed progressive growth during pre-training using a multi-stage, multi-axis growth schedule. Panigrahi et al. (2024) introduced layer dropout, progressively reducing the number of dropped layers during training. Yano et al. (2024) developed STEP, which begins pre-training with a small model and gradually increases its size in stages. Our work may be viewed as proposing another dynamic parameter schedule. However, it differs by focusing on compute-optimal strategies that gradually *reduce* model size, optimizing for inference efficiency. Since the final model is smaller, our approach leads to more efficient inference compared to methods that progressively increase model size.

3 PRELIMINARIES

Algorithm. We adapt the iterative magnitude pruning (IMP) algorithm for pre-training language models (Zhu & Gupta, 2017; Renda et al., 2020; Frankle & Carbin, 2019; Samar, 2022). We score each parameter's importance based on its magnitude. At each pruning step, we rank all model parameters globally and prune those with the lowest magnitudes. No structural constraints are imposed on the sparsity pattern. Our sparse pre-training algorithm consists of three phases:

1. *Dense training phase*: The model is trained with all parameters for N_{pre} steps;
2. *Iterative pruning phase*: The weights are being removed iteratively. Each pruning iteration starts by removing a fixed fraction of the remaining parameters, and then training for P gradient steps. This continues for N_{prune} pruning iterations, until the model reaches the desired sparsity S.
3. *Sparse recovery phase*: The sparse model is further trained with a fixed mask for N_{post} steps to recover any accuracy that is lost due to pruning.

Based on Frantar et al. (2023); Zhu & Gupta (2017); Bambhaniya et al. (2024), we fix $P = 100$.

Given a starting dense parameter count, a target sparsity S, length of each pruning iteration P, and a compute budget, we search to find the optimal allocation of compute across these three phases, N_{pre}, N_{prune} and N_{post}, to achieve the best performance in Section 6.

Effective compute. Following (Kaplan et al., 2020; Hoffmann et al., 2022), we approximate the total compute for a training run as 6 times the number of parameters multiplied by the number of training tokens. We adjust for sparsity by scaling this value linearly with respect to sparsity, following Frantar et al. (2023).

Chinchilla scaling law. Neural scaling laws model changes in final validation loss under the growth of parameters, data, compute, etc. One widely used scaling law is the Chinchilla scaling law by Hoffmann et al. (2022). This law models the relationship between the loss L, the number of parameters N, and the number of training tokens D using the following equation:

$$L(N, D) = \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + E, \tag{1}$$

where A, B, α, β, and E are free parameters: A and α describe how loss decreases with increasing model size N, while B and β describe how loss decreases with increasing training tokens D. The

constant E represents an irreducible loss. Scaling laws allow practitioners to optimize training configurations to fit within their compute budgets without running costly experiments (Sardana et al., 2024). The laws also hold theoretical value, as they capture the dynamics of how loss changes with data and parameter scaling. We present an overview of existing sparse scaling law in Appendix A.

4 METHODS

Models. We pretrain a series of models with sizes ranging from 58M to 468M parameters as reference dense models, alongside sparse models that approximately match the training compute of these dense models.

We use the LLaMA 2 (Touvron et al., 2023) base model architecture. For each unique model size, we train two versions: one using over 10x the number of tokens corresponding to Chinchilla optimal, and the other using over 20x the number of tokens.

Name	# Prunable	# Tokens
58M-10x	42M	14.7B (205x)
58M-20x	42M	29.4B (409x)
162M-10x	138M	33.5B (207x)
162M-20x	138M	67.1B (414x)
468M-10x	435M	94.4B (217x)
468M-20x	435M	188.8B (434x)

Table 1: Training details for models by size and token count. Numbers in parentheses show the token-to-prunable parameter ratio.

Table 1 provides details for each model. The model names are composed of two parts — the total number of parameters and an indicator of whether the number of training tokens exceeds 10x (but under 20x) or 20x the compute optimal value recommended by the Chinchilla scaling law (Hoffmann et al., 2022). The number of prunable parameters includes those in all linear layers, while parameters in embedding and normalization layers are left unpruned. Additionally, we report the total number of tokens used to train the dense models, followed by the ratio of training tokens to prunable parameters in parentheses.

Dataset. We use the 'en' partition of the C4 dataset (Raffel et al., 2019). Using the LLaMA 2 tokenizer, this dataset can be tokenized to 197.71 billion tokens.

Software and hardware. Our work uses TPUv4 and TPUv5 hardware for training LLMs. We modify MaxText (AI-Hypercomputer, 2024) to support sparse pre-training of LLMs.

5 SCALING ANALYSIS

Here we show that a scaling law that is of the same functional form as in eq. (1) also models the evaluation loss after sparse rather than dense pre-training when the parameter count variable is replaced with an average parameter count instead. Below we present the scaling law and an analytical argument that leads to it, and then validate this updated scaling law empirically.

5.1 A UNIFIED SCALING LAW THAT MODELS DENSE AND SPARSE PRE-TRAINING

Let T denote the total number of training iterations, where each iteration consists of a pruning step, followed by training at that fixed sparsity. Thus, a sparse pre-training run can be represented as a sequence $(N_1, D_1), (N_2, D_2) \ldots, (N_T, D_T)$, where N_k is the number of remaining parameters, and D_k is the number of tokens at iteration $k \in \{1, \ldots, T\}$. Let $\bar{N}$ denote the average parameter count during training, i.e., $\bar{N} = \frac{1}{T}\sum_{k=1}^{T} N_k$. When $N_k = N_{k'}$ for all k, k', we recover dense training, where the number of parameters does not change throughout training, and $\bar{N} = N_1$.

We model the relationship between the final evaluation loss L, the average number of model parameters $\bar{N}$, and the total number of training tokens using the following equation:

$$L(N, D) = \frac{A}{\bar{N}^{\alpha}} + \frac{B}{D^{\beta}} + E, \tag{2}$$

where D is the total number of training tokens, and A, B, E, α, and β are free positive parameters. Importantly, our proposed scaling law retains the same functional form as the Chinchilla scaling law

in Equation (1), but replaces the number of dense parameters N with the average parameter count $\bar{N}$. When sparsity is set to 0, our scaling law reduces to the original Chinchilla scaling law.

5.2 Theoretical Justification

We provide an analytical derivation of the average parameter scaling law in eq. (2). We demonstrate that, with certain assumptions—either standard in prior work or validated through empirical observations—our version of the scaling law can be derived analytically.

Assumptions. Our analysis rests on two key assumptions, one justified empirically and the other made in prior work:

1. Log loss decays linearly with log compute during pre-training;
2. Total compute for processing a fixed number of tokens is proportional to the number of active parameters in the model.

Assumption (1) has been shown to hold in Kaplan et al. (2020) when each stage of sparse pre-training follows the "compute optimal" regime , meaning that the model is appropriately sized to fully utilize the available compute. In our experiments, we extensively tune our sparse pre-training configurations to bring us close to this regime. In this setting, it is known that the loss L evolves as a function of the training compute C as

$$L(C) = (A/C)^{\alpha}\,, \tag{3}$$

where $L(C)$ represents the loss at compute C, and $\alpha > 0$ is a constant that governs the rate of loss decay as compute increases. The constant A may depend on specific configurations of the sparse pre-training run, such as the optimizer, sparsity level, and the training data distribution, but remains fixed for a particular sparse pre-training configuration.

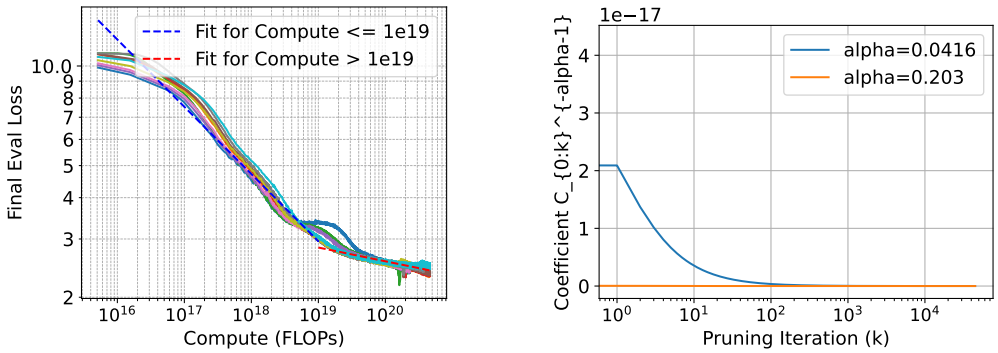


Figure 2: **Left:** Loss vs effective compute ompute for 410M models. **Right:** Estimated α coefficient in the scaling law for 410M model.

We also validate assumption (1) empirically. Figure 2 (left) shows the relationship between the training loss of sparsely pretrained 410M models and effective training compute consumption. The data confirms that the assumed predictable relationship largely holds. We empirically observe a transition point around 10^{19} floating point operations, corresponding to the first 2.6% of training steps. A linear fit on the loss data before and after this point estimates the scaling parameter α to be approximately 0.203 and 0.041, respectively.

Assumption (2) has been heavily used in previous scaling law work (Kaplan et al., 2020; Sardana et al., 2024; Frantar et al., 2023), which model compute as proportional to the number of parameters, batch size, and number of iterations. Since the number of tokens processed per step is constant, the total compute at each step is proportional only to the number of parameters in that step.

Loss modeling. To derive the average parameter scaling law, we start with a Taylor series expansion of the loss, as modelled by eq. (3) (assumption 1), around the compute point C. This yields an

approximation for the change in loss due to an increase in compute by ΔC:

$$\Delta L \approx -\alpha A^{\alpha} C^{-\alpha-1} \Delta C. \tag{4}$$

Applying the above approximation across all pruning iterations, and relying on assumption 2, we express the total changes in loss over T training steps as proportional to

$$\Delta L_{\text{total}} \propto \sum_{k=1}^{T} C_{0:k-1}^{-\alpha-1} \times N_k, \tag{5}$$

where $C_{0:k-1}$ is the accumulated compute up to step k ($C_0 = 1$), and we have used the fact that computation is known to be linear in the number of parameters (Kaplan et al., 2020) (assumption 2).

For realistic values of C and α, we find that the terms $C_{0:k-1}^{-\alpha-1}$ remain very stable. In Figure 2 (right), we plot $C_{0:k-1}^{-\alpha-1}$ as a function of pruning iterations for the 410M model experiments, with α empirically set to 0.041 and 0.203. After about 100 pruning iterations, we observe that this coefficient becomes essentially constant. Further, note that this sum of loss decreases does not take into account the increase in loss due to pruning at a specific step. This is justified as we have observed empirically that, during training, the loss spikes at the pruning step do not not effect the final loss during the pruning-training iteration. Instead, the loss only depends on the number of non-zero parameters and on the amount of computation during the iteration.

Finally, summing across iterations, we obtain that the total change in loss ΔL across all T iterations is proportional to the average number of active parameters throughout sparse pre-training, which matches our original claim.

5.3 Empirical Analysis

In this subsection, we fit our proposed scaling law with empirical data.

Fitting method. We optimize key aspects of our sparse pre-training configuration, including the learning rate, batch size, and compute allocations across the three stages of sparse pre-training (see Section 6 for details). We fit our proposed scaling law (eq. (2)) using the final evaluation loss obtained from sparse pre-training experiments with these optimal configurations. Our experiments cover 5 sparsity levels (0%, 20%, 40%, 60%, 80%), 3 model sizes (58M, 162M, 468M), and 2 training durations (10x Chinchilla optimal and 20x Chinchilla optimal), producing 30 data points.

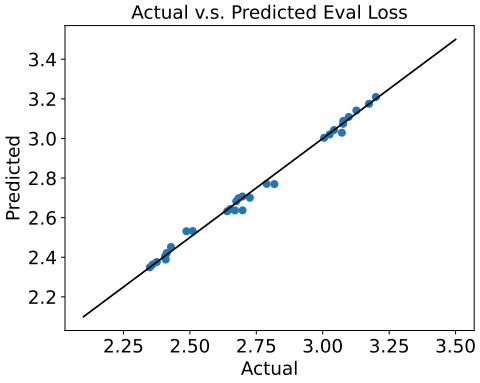


Figure 3: Predicted eval loss from our fitted scaling law versus the actual achieved final loss.

Following the methodology in Hoffmann et al. (2022), we used the L-BFGS algorithm (Liu & Nocedal, 1989) and a Huber loss with $\delta = 1 \times 10^{-3}$ to improve robustness against outliers. We set the maximum number of L-BFGS iterations to 1000, which we empirically find suitable for ensuring convergence. We initialized the scaling law's free parameters with the same random values as in Hoffmann et al. (2022). To account for possible local minima, we sampled 100 initializations from the random grid and selected the parameters with the best Huber loss, following the precedent in Hoffmann et al. (2022); Frantar et al. (2023).

Results. We present the predicted model evaluation loss and the actual final evaluation loss in Figure 3. Across different model sizes and training durations, our fitted scaling law models the final model loss with sufficient accuracy. Specifically, the average absolute difference between the predicted and actual loss is 0.016. The distribution of prediction error varies across sparsity levels: the maximum mean absolute difference occurs at 60% sparsity with 0.03, while the minimum occurs at 0% sparsity with 0.007. We attribute this disparity to the scaling analysis not fully accounting for the regularization effects of sparsity (Jin et al., 2022).

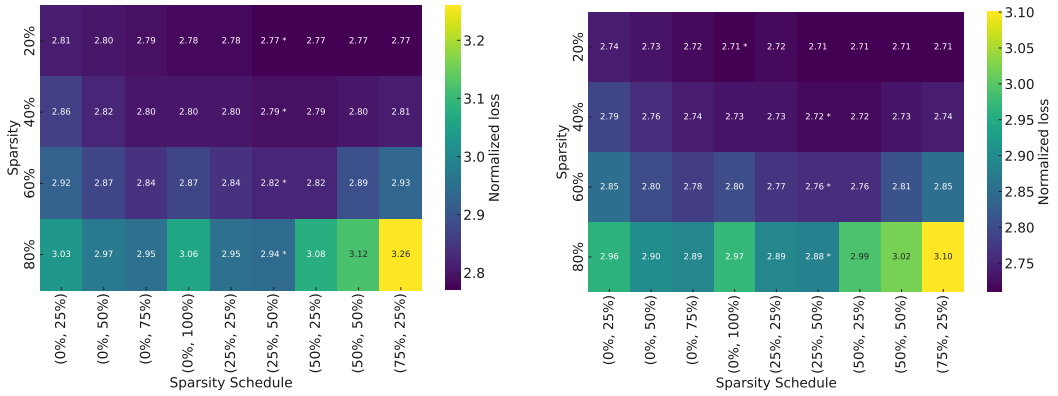


Figure 4: Optimal sparsity schedule sweep for 162M-10× (left) and 162M-20× (right) models. Each tuple on the x-axis, (t_d, t_s), represents the percentage of training time spent for dense traning (t_d), and percentage of time spent gradually pruning (t_s).

Conclusion. By replacing the total parameter count with the average parameter count in Chinchilla scaling law (eq. (1)), we demonstrate that this modified scaling law, presented in eq. (2), can effectively predict the final evaluation loss of sparse-pretrained language models.

6 Searching for the Optimal Sparse Pre-training Configuration

Recall that sparse pre-training has three stages: dense training, iterative pruning, and sparse recovery phases (Section 3). Balancing the compute resource allocation among them is crucial for effective sparse pre-training. In this section, we present experimental results from our search for the optimal sparse pre-training configurations used to derive the scaling law in section 5.

6.1 Sparse pre-training configuration sweep.

Sparsity schedule sweep. Given a dense model starting parameter count, a target sparsity, and an effective compute budget, we optimize how much compute to invest in each of the three phases of training (section 3) to find the best performing sparse network. Both empirical and theoretical studies suggest that pruning too early can trap the model in suboptimal minima (Gale et al., 2019; Frankle et al., 2020; Paul et al., 2022; Bambhaniya et al., 2024), indicating the need for substantial compute investment in the dense training phase. Likewise, removing weights too rapidly can degrade pruning outcomes (Renda et al., 2020), highlighting the importance of allocating sufficient compute to the iterative pruning phase. Balancing these factors in the design of a sparsity schedule is therefore a complex challenge.

Focusing on the 162M-10× and 162M-20× models, we systematically search for the optimal sparsity schedule by evaluating all valid combinations of compute allocated to the dense training phase (0%, 25%, 50%, 75% of total training FLOPs) and the weight removal phase (25%, 50%, 75%, 100%). For a schedule to be valid, the combined compute for the dense training and weight removal phases must not exceed 100%, with the remaining compute allocated to the sparse recovery phase. For this sweep, we adopt the same hyperparameter configurations (learning rate, batch size, etc.) used for equivalent model configurations in the Pythia suite of models (Biderman et al., 2023).

Sparsity Schedule Results. We present the results of our sparsity schedule sweep in Figure 4. We encode each schedule on the x-axis with a 2-tuple. The first value represents the proportion of compute allocated to the dense training phase, and the second value represents the compute allocated to the weight removal phase. Our results consistently show that allocating 25% of total compute to the dense training phase and 50% to the weight removal phase yields either the optimal training loss or a result within 0.01 of the minimum.

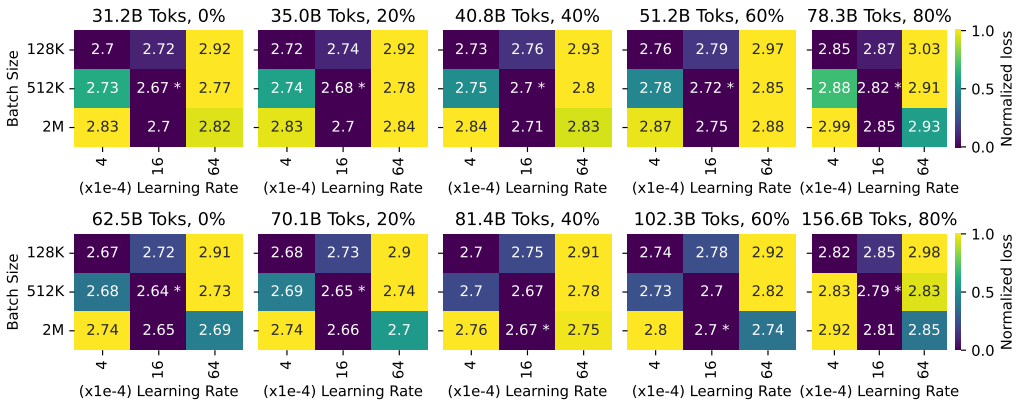
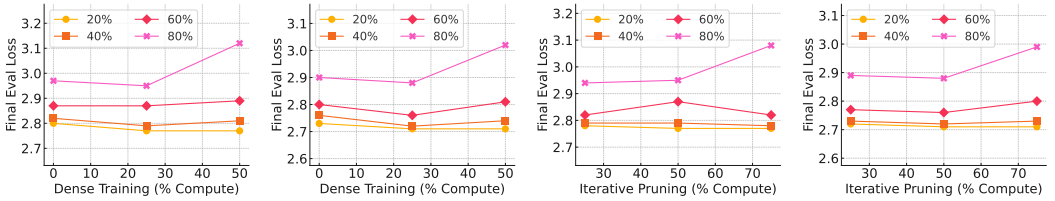


Figure 5: Batch size and learning rate sweep for 162M models.

Learning rate and batch size sweep. We subsequently investigate the optimal learning rate and batch size to use for sparse pre-training. We sweep through a grid of $[0.0004, 0.0016, 0.0064]$ for learning rate and $[0.125\text{M}, 0.5\text{M}, 2\text{M}]$ for batch size.

Learning rate and batch size results. We present our learning rate and batch size sweeps in Figure 5. Our findings consistently show that using the optimal hyper-parameters for dense pre-training either achieves the optimal evaluation loss for sparse pre-training or comes very close (within 0.01 difference).

6.2 Closer Look at Pruning Schedule



(a) Pruning Start (10x) (b) Pruning Start (20x) (c) Pruning Duration (10x) (d) Pruning Duration (20x)

Figure 6: A closer look at failure modes for non-optimal sparsity schedules.

In this section, we visualize the impact of allocating different fractions of the total compute to the dense training and iterative pruning phases on final evaluation loss.

Dense training compute. To determine the optimal fraction of compute to allocate to the dense pre-training phase, we vary the dense compute between 0%, 25%, and 50%. Throughout these experiments, we keep the iterative pruning phase compute fixed at 50%, as this was previously found to be optimal, and adjust the sparse recovery phase compute to maintain a constant total compute for sparse pre-training. The results are shown in Figures 6a and 6b.

Our findings indicate a clear optimal allocation of dense training compute at 25% of total compute, where 7 out of 8 cases reach their lowest loss. This trend is consistent across different sparsity levels (ranging from 20% to 80%) and both training regimes (10x and 20x Chinchilla-optimal).

Additionally, the results suggest that allocating too much compute to dense training (50%) leads to worse final loss, particularly for high-sparsity models. This underscores the importance of allocating sufficient compute to the sparse recovery phase.

Iterative pruning compute. We explore the effect of varying the compute allocation to the iterative pruning phase. First, we fix the dense training compute at 25%, as this was previously found to be

optimal. Then, we vary the pruning duration while adjusting the sparse recovery phase to keep the total compute constant. We visualize these results in Figures 6c and 6d.

We find that the optimal allocation of iterative pruning compute varies across training regimes. For the 10x Chinchilla model, the lowest evaluation loss occurs with a 25% allocation to iterative pruning (2.83), while for the 20x Chinchilla model, the optimal allocation is 50% (2.77). Despite these differences, allocating 50% of total compute to iterative pruning consistently results in reasonable evaluation loss across both regimes. Interestingly, extending the iterative pruning allocation beyond 50% tends to degrade performance, particularly in high-sparsity models (80%). These findings suggest that high-sparsity models benefit most from moderate allocations to iterative pruning, ensuring sufficient compute for sparse recovery after weight removal.

7 Concluding remark

Our work examines pre-training large language models with parameter sparsity and presents a unified scaling law that effectively models both sparse and dense scaling.

Value of sparse pre-training. Our scaling analysis implies that sparsely pre-trained language models achieve a similar final evaluation loss to smaller dense models with the same average active parameter count, as validated directly in Figure 1. When compared to their matching dense pre-training configurations, sparse pre-training shifts the active parameter count across training steps. Specifically, sparse pre-training begins with a higher active parameter count early in training and reduces it later, while dense pre-training keeps the active parameter count constant throughout. Importantly, one may design this shift (as we did in our work) without changing the total effective training compute. Within the model scales, sparsity levels, and schedules we explored, this shift does not negatively affect the trade-off between training compute and final evaluation loss.

For any dense pre-training configuration, our analysis suggests that one can apply this shift in active parameter count to create a sparse pre-training configuration, with little impact on the final evaluation loss and no change to the total effective training compute. As a result of applying this shift in active parameters, the final model will be smaller than its dense counterpart, which has a size equivalent to the average active parameters of the sparsely pre-trained model throughout pre-training.

Compression rate. Our findings suggest that up to a certain compression rate, sparse pre-training effectively compresses the language model without loss in quality. This compression rate should be computed as the ratio of the average active parameter count to the final active parameter count. Within the language models we explore, the maximum compression rate is reached at 80% target final sparsity, where our sparsity schedule results in an average active parameter count of approximately 40% of total the initial dense parameter count, yielding a 2x lossless compression rate.

Limitation. We note that, due to the lack of adequate software and hardware support for executing matrix multiplications with unstructured sparsity, we are unable to demonstrate computational savings from sparse pre-training. For the same reason, the scale of our study is limited, as the actual compute required may be up to $1/(1 - \text{sparsity})$ times the effective compute we estimate for sparse pre-training. For experiments with 80% sparsity, this factor reaches 5x.

Another limitation of our work is the reliance on evaluation perplexity as a proxy for model quality. It is well known, however, that perplexity does not always correlate with model utility. As a result, we acknowledge the absence of downstream task evaluations as a limitation of this study.

In this work, we demonstrate that sparse pre-training offers a viable alternative to dense pre-training by reducing the inference costs without sacrificing the trade-off between training compute and model quality. We propose a scaling law that models both sparse and dense pre-training with a unified functional form. Our proposed scaling law, which incorporates the average number of active parameters, accurately predicts the loss of sparse-pretrained LLMs.

Reproducibility statement. We have detailed our dense LLM training configurations in Section 4, along with our hardware and software setup. Additionally, in Section 6, we provide an extensive discussion of the pruning configurations we explored and identified.

REFERENCES

AI-Hypercomputer. Maxtext: A framework for training large language models. https://github.com/AI-Hypercomputer/maxtext, 2024. Accessed: 2024-10-01.

Abhimanyu Rajeshkumar Bambhaniya, Amir Yazdanbakhsh, Suvinay Subramanian, Sheng-Chun Kao, Shivani Agrawal, Utku Evci, and Tushar Krishna. Progressive gradient flow for robust n:m sparsity training in transformers, 2024. URL https://arxiv.org/abs/2402.04744.

Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pp. 26–33, Toulouse, France, July 2001. Association for Computational Linguistics. doi: 10.3115/1073012.1073017. URL https://aclanthology.org/P01-1005.

Yamini Bansal, Behrooz Ghorbani, Ankush Garg, Biao Zhang, Colin Cherry, Behnam Neyshabur, and Orhan Firat. Data scaling laws in NMT: The effect of noise and architecture. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1466–1482. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/bansal22b.html.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023. URL https://arxiv.org/abs/2304.01373.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2021. URL https://arxiv.org/abs/1911.11134.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL https://arxiv.org/abs/1803.03635.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis, 2020. URL https://arxiv.org/abs/1912.05671.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. URL https://arxiv.org/abs/2301.00774.

Elias Frantar, Carlos Riquelme, Neil Houlsby, Dan Alistarh, and Utku Evci. Scaling laws for sparsely-connected foundation models, 2023. URL https://arxiv.org/abs/2309.08520.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019. URL http://arxiv.org/abs/1902.09574.

Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. Scaling laws for neural machine translation, 2021. URL https://arxiv.org/abs/2109.07740.

Joshua Goodman. A bit of progress in language modeling, 2001. URL https://arxiv.org/abs/cs/0108005.

Mitchell A Gordon, Kevin Duh, and Jared Kaplan. Data and parameter scaling laws for neural machine translation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5915–5922, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.478. URL https://aclanthology.org/2021.emnlp-main.478.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models, 2024. URL https://arxiv.org/abs/2402.00838.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015. URL https://arxiv.org/abs/1506.02626.

B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pp. 293–299 vol.1, 1993. doi: 10.1109/ICNN.1993.298572.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks, 2017. URL https://arxiv.org/abs/1707.06168.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Tian Jin, Michael Carbin, Dan Roy, Jonathan Frankle, and Gintare Karolina Dziugaite. Pruning's effect on generalization through the lens of training and regularization. *Advances in Neural Information Processing Systems*, 35:37947–37961, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Denis Kuznedelev, Eldar Kurtic, Eugenia Iofinova, Elias Frantar, Alexandra Peste, and Dan Alistarh. Accurate neural network pruning requires rethinking sparse optimization. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=vgthYeRBAF.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989. ISSN 1436-4646. doi: 10.1007/BF01589116. URL https://doi.org/10.1007/BF01589116.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt, 2019. URL https://arxiv.org/abs/1912.02292.

Abhishek Panigrahi, Nikunj Saunshi, Kaifeng Lyu, Sobhan Miryoosefi, Sashank Reddi, Satyen Kale, and Sanjiv Kumar. Efficient stagewise pretraining via progressive subnetworks. *arXiv preprint arXiv:2402.05913*, 2024.

Mansheej Paul, Feng Chen, Brett W Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? *arXiv preprint arXiv:2210.03044*, 2022.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. Ac/dc: Alternating compressed/decompressed training of deep neural networks, 2021. URL https://arxiv.org/abs/2106.12379.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning, 2020. URL https://arxiv.org/abs/2003.02389.

Jonathan S. Rosenfeld, Jonathan Frankle, Michael Carbin, and Nir Shavit. On the predictability of pruning across scales, 2021. URL https://arxiv.org/abs/2006.10621.

Anshul Samar. Creating sparse gpt-3 models with iterative pruning, 2022. URL https://cerebras.ai/blog/creating-sparse-gpt-3-models-with-iterative-pruning. Accessed: 2024-09-30.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws, 2024. URL https://arxiv.org/abs/2401.00448.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models, 2024. URL https://arxiv.org/abs/2306.11695.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning, 2024. URL https://arxiv.org/abs/2310.06694.

Kazuki Yano, Takumi Ito, and Jun Suzuki. Step: Staged parameter-efficient pre-training for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 607–614, 2024.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. Masked structural growth for 2x faster language model pre-training. *arXiv preprint arXiv:2305.02869*, 2023.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017. URL https://arxiv.org/abs/1710.01878.

A EXISTING SPARSE SCALING LAW

Frantar et al. (2023) propose a model for the evaluation loss of a sparsely pre-trained LLM based on the final sparsity S, the number of non-zero parameters N at the end of pre-training, and the amount of training data D, as follows:

$$L(S, N, D) = \left(a_S(1-S)^{b_S} + c_S\right) \cdot \left(\frac{1}{N}\right)^{b_N} + \left(\frac{a_D}{D}\right)^{b_D} + c$$

Compared to the Chinchilla scaling law, this model introduces the following modifications:

1. A sparsity term $a_S(1-S)^{b_S}$, which introduces two new sparsity-specific parameters, a_S and b_S, allowing the model to account for the effect of sparsity on loss.
2. Instead of using the total number of parameters N as in dense training, the model uses the number of non-zero parameters remaining at the end of sparse pre-training.