Just CHOP: Embarrassingly Simple LLM Compression

Anonymous ACL submission

Abstract

Large language models (LLMs) enable unparalleled few- and zero-shot reasoning capabilities but at a high computational footprint. A growing assortment of methods for compression promises to reduce the computational burden of LLMs in deployment, but so far, only quantization approaches have been demonstrated to be effective for LLM compression while maintaining zero-shot performance. A critical step in the compression process, the pretrain-then-finetune paradigm, has largely been overlooked when adapting existing pruning strategies to LLMs or proposing new ones. In this work, we show that embarrassingly simple layer pruning coupled with an extended language model pretraining as the finetuning phase produces state-of-the-art results against structured and even semi-structured compression of models at a 7B scale while being more inference efficient. We call this method LayerChop, where we deterministically remove layers from a model followed by task-agnostic finetuning of the remaining weights by continued self-supervised pretraining. At this scale, we also show how distillation, which has been super effective in task-agnostic compression of smaller BERT-style models, becomes inefficient against our simple pruning technique. We release our code and evaluation setup to facilitate reproducibility and advancement of task-agnostic compression for LLMs.[1]

1 Introduction

Large language models (LLMs) have demonstrated efficacy in a wide range of real-world tasks mainly due to their few- and zero-shot reasoning capabilities (Brown et al., 2020). However, these successes come at a high computational cost of training, and operational constraints on memory and throughput demand effective compression methods for LLM deployment.

While a flurry of methods for compressing LLMs have emerged (Sun et al., 2023; Kim et al., 2024) since the shift from small task-specific models to transfer learning large pretrained models, the vast majority of compression methods have been developed for the BERT-style, non-autoregressive (encoder-only) pretrain-then-finetune paradigm (Sanh et al., 2019; Liang et al., 2023). This differs substantially from the autoregressive (decoder-only) models that enable zero-shot reasoning via in-context learning. Aside from model architecture, the most notable difference is in the availability and scale of training data (Soldaini et al., 2024).

The majority of compression methods have been developed for the *task-specific* setting (Voita et al., 2019; McCarley et al., 2019; Michel et al., 2019), where the compressed model has access to supervised training data for task-specific finetuning or selection of parameters; these approaches are fundamentally incompatible with compressing an LLM for general-purpose in-context learning. Methods have also been developed for *task-agnostic* BERT-style LM compression (Chen et al., 2020b; Fan et al., 2019), where the goal is to remove model parameters while maintaining language modeling perplexity. While task-agnostic compression is more compatible with the current paradigm, most BERT-style methods assume that training on the entire pretraining corpus is feasible (Sanh et al., 2019; Chen et al., 2020b; Liang et al., 2023). This assumption is no longer valid for LLMs.

As a result, unlike their non-autoregressive predecessors, methods for compressing autoregressive LLMs have focused on reducing model size with no subsequent finetuning, even on task-agnostic pretraining data. Under this constrained setting, methods compressing model parameters and activations via quantization have shown the most success to date (Dettmers et al., 2022), and although distillation and pruning methods have the potential to

[1] https://github.com/anonymous

provide orthogonal benefits alongside model quantization, those benefits still need to be realized.

In this work, we demonstrate the utility of task-agnostic finetuning for compression of autoregressive LLMs. We explore two simple structured pruning techniques followed by continued pretraining under a limited token budget: LayerCHOP and DimCHOP. LayerCHOP reduces model depth by deterministically removing entire model layers. DimCHOP reduces model width by uniformly removing the same number of dimensions from each weight matrix in the network by adapting a parameter-level pruning criterion proposed in PLATON (Zhang et al., 2022) to dimensions. Furthermore, under the relaxed assumption that the compression procedure may include additional gradient updates on the pretraining data, we demonstrate that simply pretraining a smaller model from scratch on a limited budget of tokens represents a strong baseline against which we can compare compression methods. Finally, inspired by the success of augmenting task-agnostic compression with distillation objectives (Liang et al., 2023) we test this hypothesis for LLMs in the new pretrain-then-finetune paradigm.

Evaluation on language model perplexity and on a suite of 6 zero-shot tasks across three model sizes, we demonstrate the following:

- LayerCHOP coupled with a finetuning phase, outperforms existing structured and even semi-structured compression at the 7B scale.
- Pretraining smaller models from scratch forms a strong baseline against which structured compression with a finetuning phase can be compared against.
- Both LayerCHOP and DimCHOP also outperform their distillation objective augmented variants to show that distillation does not scale efficiently in large model and data regimes.

2 Related Work

Pruning seeks to identify sub-networks within larger models that yield the best possible performance with increased computational efficiency. Pruning uses heuristics (Sanh et al., 2020; Li et al., 2021a) to identify parts of a model to prune.

Lottery ticket hypothesis (Frankle and Carbin, 2019) introduces the concept of sparse sub-networks within a model, called winning tickets, which have the full model's task capability, but pruning of weights is unstructured. A different line of work introduces semi-structured pruning (Kao et al., 2022), which improves throughput speed over unstructured pruning via accelerator-specific kernels. Structured pruning approaches (Fan et al., 2019; Kao et al., 2022) aim to prune larger blocks within a model while considering architecture nuances for improving inference speed. Structured pruning can be coarse- or fine-grained. Coarse-grained pruning removes entire model layers (Fan et al., 2019). In the case of LMs, fine-grained pruning methods prunes atomic components like attention-heads (Voita et al., 2019; Michel et al., 2019; Li et al., 2021b) or hidden layers (Hou et al., 2020; Chen et al., 2020b; McCarley et al., 2019).

Another classification approach for pruning is based on task-agnostic vs. task-specific pruning. Task-agnostic pruning approaches (Chen et al., 2020b; Fan et al., 2019) prune a model on pretraining data and then add task specialization as a second step. Task-specific pruning (Voita et al., 2019; McCarley et al., 2019; Michel et al., 2019) methods specialize their models on end-task data during the pruning process.

More recently, pruning has been extended to larger decoder-only LLMs with semi-/unstructured (Sun et al., 2023; Frantar and Alistarh, 2023) and structured pruning methods (van der Ouderaa et al., 2023; Ashkboos et al., 2024; Kim et al., 2024; Xia et al., 2023).

Knowledge Distillation can be similarly divided between task-specific (Sun et al., 2019; Turc et al., 2019; Mukherjee et al., 2021; Tang et al., 2019; Xia et al., 2022) and task-agnostic methods (Sanh et al., 2019; Jiao et al., 2019; Sun et al., 2020; Wang et al., 2020b,a; Liang et al., 2023) for BERT-style models. Both task-specific and task-agnostic distillation methods finetune BERT-style encoder models on end tasks to evaluate them. Seq-KD (Kim and Rush, 2016) proposed task-agnostic distillation with a joint word level and sequence level objective for seq2seq models (Vaswani et al., 2017), which has been expanded to larger models like Distil Whisper (Gandhi et al., 2023).

Some methods follow pruning with a distillation phase (Hou et al., 2020; McCarley et al., 2019). More recently, methods like Co-Fi (Xia et al., 2022) and Homotopic-distillation (Liang et al., 2023) combine pruning with distillation by applying distillation losses during the pruning process.

Method	Encoder-only	Decoder-only	This work
Struct. depth	Poor Man's BERT (Sajjad et al., 2020)	ShortenedLlama (Kim et al., 2024)	LayerCHOP
Struct. width	PLATON (Zhang et al., 2022)	SliceGPT (Ashkboos et al., 2024); LLMSurgeon (van der Ouderaa et al., 2023)	DimCHOP
Un-/Semi-struct.	LTH-BERT (Chen et al., 2020a)	Wanda (Sun et al., 2023); SparseGPT (Frantar and Alistarh, 2023)	—
Distillation	Homo-Distil (Liang et al., 2023)	—	CHOP+distill
Training data	Pretrain. (GBs)	Few batches	Pretrain. (TBs)

Table 1: Taxonomy of compression methods. We adapt pruning strategies from BERT-style models to LLMs under the pretrain-then-finetune paradigm. We also provide a list of recent or concurrent works for LLMs which compress them but do not follow it with a finetuning phase. We do not consider an un- or semi-structured form of CHOP as our early experiments validated our structured methods as having sufficient end-task accuracy while maintaining superior inference efficiency.

3 How to Train Your (Compressed) LLM

This section describes methods for task-agnostic model compression of large language models (LLMs). From previous literature, we adapt pruning strategies that are simple, efficient, and go well with continued pretraining on a large data corpus described in Section 4.1.

We prune models in two simple ways: model depth (LayerCHOP; §3.1) and model width (DimCHOP; §3.2). It has been observed in the BERT model compression literature that adding distillation in conjunction with pruning amplifies the result of compression (Liang et al., 2023). Hence, we experiment with the same for decoder-only LLMs in the large data regime. A taxonomy of our adapted methods and where they lie within the current compression literature is summarized in Table 1.

3.1 LayerCHOP

LayerCHOP is a layer removal pruning strategy for decoder-only transformer models for reducing model depth. We remove 50% of layers from the pretrained Transformer architecture, corresponding to removing $\sim$ 50% of parameters. We continue (i.e., resume) training each model with a language modeling loss on data sampled from a similar corpus to the respective model's pretraining corpus (§4.1). Choosing where and when to prune layers is a critical design decision to maximize performance recovery after pruning. We experiment with five configurations for layer pruning shown in Figure 5 in Appendix C. We always retain the first and the last layers as they interact with the embedding table and the final feed-forward layer of the model. Considering when to enact pruning primarily concerns either pruning all layers at initialization or incrementally removing layers periodically during the continued training of the model.

Our layer-pruning strategy is similar to that in Poor Man's BERT (Sajjad et al., 2020). However, aside from the architectural differences between models (encoder-only vs decoder-only), our work contrasts in that we compress and evaluate in a stricter task-agnostic zero-shot setting without access to task-specific tuning. LayerDrop (Fan et al., 2019), similarly proposes *layer dropout* as layer removal for compression. However, a key difference to our work is that layer dropout critically requires training from scratch with randomly masked layers to facilitate successful inference after pruning. Our method removes this constraint for broader utility—our technique can be applied to adapt and compress any publicly available LLM checkpoint. More recently, Shortened Llama (Kim et al., 2024), has proposed layer pruning and then training LoRA (Hu et al., 2021) weights on top of the pruned model. The main difference in their work is that they do not train their LoRA parameters at a pretraining corpus scale.

3.2 DimCHOP

Our second method is defined for structured compression of decoder-only models in their width, i.e., dimensions, and we call it DimCHOP. This method is adapted for decoder-only LLM models from PLATON (Zhang et al., 2022), a technique proposed for parameter pruning of encoder-only BERT models. For this method, we retain the same number of layers in the model as the original uncompressed version, but each layer has some dimensions with weights set to zero, e.g., for some layer $W \in \mathbb{R}^{m \times n}$ we apply mask $M \in \mathbb{R}^{m \times n}$, where $M_n = 0$ and we take the element-wise Hadamard product $W \odot M$. We iteratively mask dimensions in weight matrices across the model

in a balanced fashion, i.e., we zero out the same number of dimensions in each weight matrix of the model. This iterative pruning of dimensions is carried out as we continue training the model on language modeling loss on data sampled from a corpus similar to the pretraining set (§4.1).

The pruning criterion defined for selecting dimensions follows PLATON's sensitivity score $\mathcal{I}_j$ for each model parameter j at some time during training t. Score $\mathcal{I}_j^t$ is the magnitude of the gradient-weight product (Eq. 1). PLATON further introduces an uncertainty score $\mathcal{U}_j$ (Eq. 3) for each parameter, representing the absolute difference between local sensitivity and an exponential weighted average parameter sensitivity (Eq. 2). The final importance score (Eq. 5) is the Hadamard product between the exponential moving averages of the sensitivity and the uncertainty scores.

$$\mathcal{I}_j^t = |\theta_j^T \nabla \mathcal{L}(\theta)| \tag{1}$$
$$\bar{\mathcal{I}}_j^t = \beta_1 \bar{\mathcal{I}}_j^{t-1} + (1-\beta_1)\bar{\mathcal{I}}_j^t \tag{2}$$
$$\bar{\mathcal{U}}_j^t = |\mathcal{I}_j^t - \bar{\mathcal{I}}_j^t| \tag{3}$$
$$\bar{\mathcal{U}}_j^t = \beta_2 \bar{\mathcal{U}}_j^{t-1} + (1-\beta_2)\bar{\mathcal{U}}_j^t \tag{4}$$
$$\mathcal{S}_t = \bar{\mathcal{I}}_j^t \odot \bar{\mathcal{U}}_j^t \tag{5}$$

While PLATON limits the importance scores to each parameter, we compute the importance scores for a dimension $\mathcal{I}_d$ (Eq. 6) by taking an L_1 norm over the importance scores of the parameters in the column of the weight matrix. The importance scores of all dimensions within a weight matrix are then sorted and the dimensions with the worst $k\%$ of the importance scores are set to 0 using the mask M. The $k\%$ of dimensions being set to 0 is defined by an exponential schedule which increases the number of dimension being pruned as the training progresses.

$$\bar{\mathcal{S}}_d^t = ||\bar{\mathcal{S}}_{[:,j]}^t||_1 \tag{6}$$

The difference between PLATON and our work is that PLATON prunes BERT-style models on specific end-task data at a parameter level, whereas we extend the idea of pruning parameters to pruning dimensions for decoder-only LLMs in a task-agnostic setting. This is more similar to Homotopic-Distillation (Liang et al., 2023), which prunes dimensions from a BERT-style model, but augments the pruning with distillation objectives.

3.3 Distillation-augmented Pruning

Homotopic distillation Liang et al. (2023) uses the PLATON pruning criterion discussed in §3.2. This technique aggregates the importance score over weight matrix columns to prune an equal number of model dimensions from each weight matrix in a model. Liang et al. (2023) identifies that augmenting iterative dimension pruning with distillation-based continued training produces better compression outcomes in a task-agnostic setting. We evaluate a similar hypothesis in our structured compression setup of decoder-only models evaluated in a zero-shot fashion, and augment LayerCHOP and DimCHOP with distillation objectives. Eq. 7 gives the combined distillation objective for this setup comprising of the language modeling objective; KL-divergence over the outputs of the teacher and student; and mean-squared error distillation objective over attention outputs, intermediate layer representations, and the embedding table.

$$\mathcal{L}_\Sigma = \mathcal{L}_{\text{LM}} + \mathcal{L}_{\text{distill}} + \mathcal{L}_{\text{hid}} + \mathcal{L}_{\text{att}} + \mathcal{L}_{\text{emb}} \tag{7}$$
$$\mathcal{L}_{\text{hid}}(\theta_s, \theta_t) = \sum_{k=1}^{K} ||H_t^k, H_s^k W_{\text{hid}}^k||_2^2 \tag{8}$$
$$\mathcal{L}_{\text{attn}}(\theta_s, \theta_t) = \sum_{k=1}^{K} ||A_t^k, A_s^k||_2^2 \tag{9}$$
$$\mathcal{L}_{\text{emb}}(\theta_s, \theta_t) = ||E_t, E_s W_{\text{emb}}||_2^2 \tag{10}$$

Eq. 8 defines the alignment between the intermediate representations of the model at each layer. We introduce a learned linear projection, $W_{\text{hid}}^k \in \mathbb{R}^{t \times s}$, to match the dimension between student and teacher representations. Eq. 9 defines the mean squared error between attention outputs at each layer. Eq. 10 defines the same loss between the embedding tables of the student and teacher models, with a learnable weight matrix $W_{\text{emb}} \in \mathbb{R}^{s_{\text{emb}} \times t_{\text{emb}}}$ matching dimensions between embeddings.

4 Setup

4.1 Experimental setup

All our experiments use the C4 (Raffel et al., 2020) dataset for pretraining or pruning of models except for the pretrained llama2-7B (Touvron et al., 2023). C4 consists of approx. 160B web-crawled tokens. Our best-performing and most efficient compression strategy, LayerCHOP (Section 3.1), trains on 1 complete epoch of C4, i.e., token budget of 160B

tokens, for models at all scales. Other ablations (Section 3.2, 3.3) train on smaller token budgets from the C4 training set. For language modeling evaluation, each experiment in the paper uses at least 13.2M C4 tokens from a separate validation set as we use 6400 batches at the full sequence length of the models used.

Our experiments use models at 3 scales: 300M, 1.1B, and 7B. For our 7B model pruning experiments, we use llama2-7B (Touvron et al., 2023). For the other two sizes, we train our own models on 1 epoch of C4 training set with 300M and 1.1B parameters. These smaller models follow PaLM (Chowdhery et al., 2022) like architecture with parallel attention and feed-forward network (FFN) blocks, SwiGLU activation (Shazeer, 2020), and fused FFN layers. All models in our experiment use flash-attention (Dao et al., 2022) for efficiency. We also train half-sized models for each scale, with half the number of layers as the base model, for pretraining comparisons in Section 5.2. The architecture choices are summarized in Table 4 in Appendix A.

For experiments with the 300M and 1.1B models, we use the Lion optimizer (Chen et al., 2023) with β set to $(0.9, 0.95)$. Weight decay is 1×10^{-4} but is omitted for bias and normalization parameters. We found some stability issues with Lion at larger model sizes and reverted back to the more common AdamW (Loshchilov and Hutter, 2017) for our llama2-7B runs. The values of β parameters are the same, while the weight decay is 0.1. Each pruning and distillation experiment uses the same learning rate as pretraining, which is mentioned in Table 4 in Appendix A. The learning rate is warmed up to this value in 2000 steps and then decayed using a cosine schedule to 0.1 times the peak value. For our compression experiments, we start with a model checkpoint and train 500 steps before any compression is applied to a model to accumulate optimizer states.

4.2 Evaluation setup

We evaluate our model on 6 tasks from 2 categories: common sense reasoning and science question answering. All tasks are evaluated in a zero-shot setting by providing the language model with a prompt from Eleuther-AI evaluation harness (Gao et al., 2021) and a possible completion. We score the model output for each completion. The completion with the highest likelihood is the prediction to compute task accuracy. The completion likelihood can be normalized by the character count in the completion (Gao et al., 2021, length normalized accuracy). Table 5 in Appendix B lists tasks in each evaluation category and respective metrics.

5 Compression Results

The results section makes comparisons between task performance and efficiency of compression and inference for different variants of task-agnostic model compression. All results use the best configuration of LayerCHOP(Appendix. C) and DimCHOP (Appendix. D) which is discussed in detail in the Appendix.

5.1 Comparison against structured and unstructured pruning methods

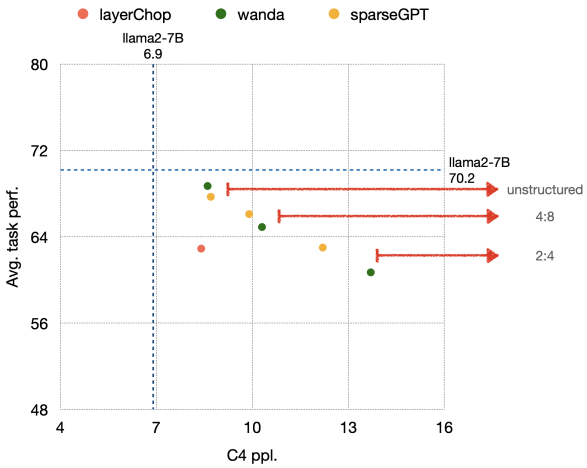

Figure 1: Comparison between LayerCHOP and un-/semi-structured compression of llama2-7B from Wanda (Sun et al., 2023) and SparseGPT (Frantar and Alistarh, 2023). Both Wanda and SparseGPT make their models public and for fair comparison we evaluate them on end tasks in our eval setup. We can see layerCHOP being better than 2:4 semi-structured pruned models and only marginally worse than 4:8 pruned models.

This section compares the results of LayerChop to concurrent unstructured/semi-structured and more recently proposed structured compression methods for LLMs. The other techniques we compare here do not continue pretraining their models on data after pruning. When combined with continued pretraining, we show how the simple pruning strategy of layer removal outperforms complicated pruning heuristics from other methods.

In Figure 1, we compare LayerChop to some of the recent un-/semi-structured pruning approaches for decoder-only LLMs. This comparison is unusual because these two are in entirely dif-

method	**shortenedLlama**	**sliceGPT**	**LLMSurgeon**
# common eval tasks	5	4	5
base model	llama-7B	llama2-7B	llama2-7B
method base avg./method pruned avg.	68.9/57.9	74.7/56.6	68.6/46.6
% model pruned/% task score decrease	35%/-16%	30%/-24.2%	50%/-32.1%
our base avg./our pruned avg.	66.2/58.2	72.0/62.8	66.2/58.2
our % model pruned/our % task score decrease	50%/**-12.1%**	50%/**-12.8%**	50%/**-12.1%**

Table 2: Comparison against the concurrent works of Shortened Llama (Kim et al., 2024), SliceGPT (Ashkboos et al., 2024), and LLMSurgeon (van der Ouderaa et al., 2023). Since each paper has its own eval suite and base model, and not all of these methods have publicly available checkpoints, we compare them by averaging the base model scores and pruned model scores on end tasks which are common with our method. Then we find the percentage of end tasks score decrease with the percentage of model being pruned and list them in this table. As we can, LayerCHOP prunes the model to 50% of the size, better than the other methods other than LLMSurgeon, and has the least reduction in average end task score across all methods.

ferent classes of pruning techniques. However, we posit LayerChop against 2:4 semi-structured pruning methods, Wanda (Sun et al., 2023) and sparseGPT (Frantar and Alistarh, 2023), because 2:4 pruned models improves speed on NVIDIA hardware using specialized kernels. We show that with continued pretraining LayerChop outperforms 2:4 semi-structured pruning on llama2 from Wanda and sparseGPT. Additionally, Table 3 highlights the end-to-end inference speed gained by these methods. LayerChop incurs additional training costs due to continued pretraining but makes up the difference by being 1.84x faster than the base 7B llama2 model against 1.24x end-to-end speed achieved by the 2:4 pruned model. Given a training token budget for LayerChop, we can calculate the number of inference queries it will take to break even compared to the 2:4 pruning approach, which is one-shot pruning.

Table 2 highlights how LayerChop compares to some of the recently proposed structured pruning methods for larger decoder-LLMs. Shortened Llama (Kim et al., 2024) is a layer pruning method that trains LoRA (Hu et al., 2021) weights after pruning, while SliceGPT (Ashkboos et al., 2024) and LLMSurgeon (van der Ouderaa et al., 2023) are dimension pruning approaches that prune model dimensions based on a heuristic. Since the base model and the evaluation setup are different across our work and these papers, we highlight the overlapping number of tasks where results were presented in each of these works, and compute the average of the reported accuracies for the base model and the final pruned models only on the tasks that overlap. We propose a percentage decrease in average task accuracy to measure each pruning strategy's effectiveness. As we can see, LayerChop consistently outperforms all other heuristic-based structured pruning approaches while pruning 50% of the model, a pruning rate which is equaled only by LLMSurgeon (van der Ouderaa et al., 2023).

5.2 Comparisons against simple pretraining baselines

We take the best configuration (Appendix C) of our most competitive pruning strategy, i.e., LayerCHOP, and continue pretraining for an extended token budget of 160B tokens from C4 corpus to compare against pretraining a similar-sized model from scratch at 3 models scales: 300M, 1.1B, and 7B. We present the average task accuracies over the evaluation suite defined in Section 4.2 and language modeling perplexity on the C4 eval set defined in Section 4.1 in Figure 2.

As we see in both the task average and perplexity trends, pruning with continued training eventually converges towards pretraining at the 300M and 1.1B scale after a certain number of tokens. At the

compression	**2:4**	**LayerCHOP**
Uncompressed latency	312ms	351ms
Compressed latency	251ms	191ms
Speed	1.24x	1.84x
Breakeven query	240M	360M

Table 3: Comparison between end-to-end inference speeds of 2:4 semi-structured pruned models and LayerCHOP. The numbers for 2:4 pruning are taken from Wanda (Sun et al., 2023). LayerCHOP, a form of structured pruning, is 1.84x faster than the base model, against 1.24x speed improvement of 2:4 pruned model. The additional cost of training LayerCHOP can be subsumed by the gain in inference speed in the breakeven query number given for each type of pruning.

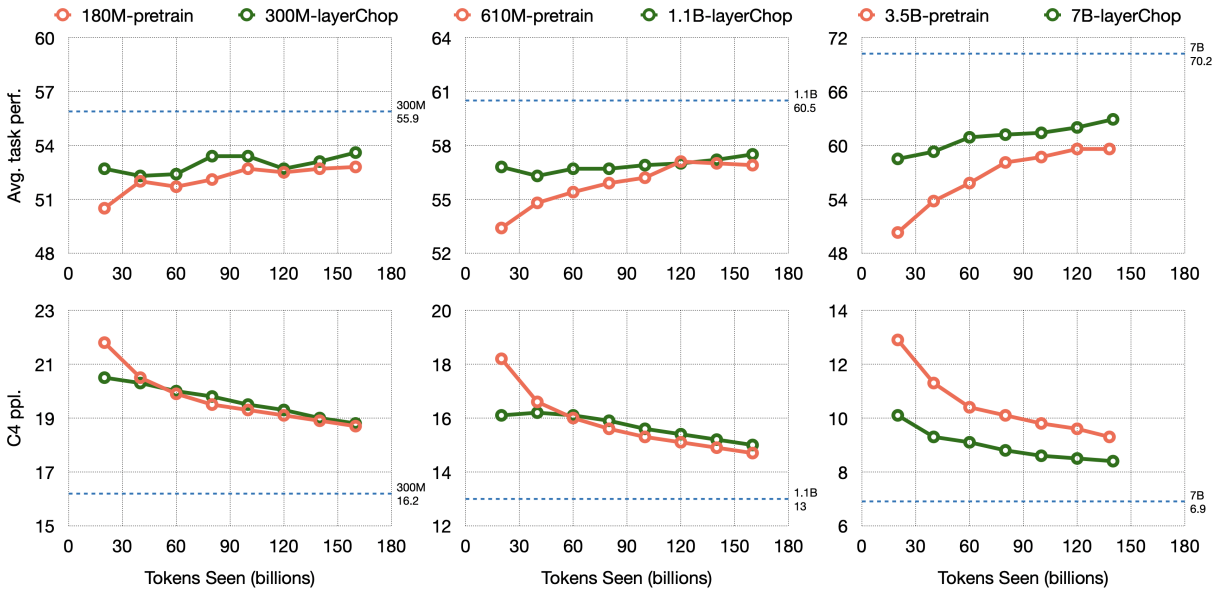

Figure 2: Language modeling and average task accuracy performance plotted against token budget for LayerCHOP against pretraining baselines. We observe that for smaller 300M and 1.1B models, pruning and finetuning of models converge to pretraining baselines. However, this trend is not obvious at the 7B scale.

same time, it looks to move towards convergence at the 7B scale. For coarse-grained model pruning, this trend deviates from the idea of "train large and then compress", as we show in these experiments that if the token budget is large enough, it might be a better idea to pretrain models from scratch instead of pruning them in a coarse fashion. Note that this trend contrasts what pretraining with fine-grained pruning observes in Sheared Llama (Xia et al., 2023).

5.3 Comparing LayerCHOP and DimCHOP

In this section, we compare the results of LayerCHOP and DimCHOP in terms of performance and efficiency. Figure 3 shows how both methods perform in the same ballpark on average task performance and language modeling after consuming the same number of pretraining tokens; however, the algorithmic differences in the two pruning strategies make LayerCHOP much more efficient to train compared to DimCHOP. LayerCHOP instantly prunes half the number of layers from a model and continues pretraining only half-sized models. At the same time, DimCHOP is an iterative algorithm that prunes dimensions after every few steps. However, it further trains the zeroed-out dimensions before re-evaluating them for pruning based on importance scores (Section 3.2). We provide more details on LayerCHOP and how its hyperparameters were selected in Appendix D.

5.4 Efficiency of distillation with a large pretraining corpus

Figure 4 compares average task performance and language modeling perplexity results for LayerCHOP and DimCHOP against distillation augmented versions of the algorithms described in Sec-

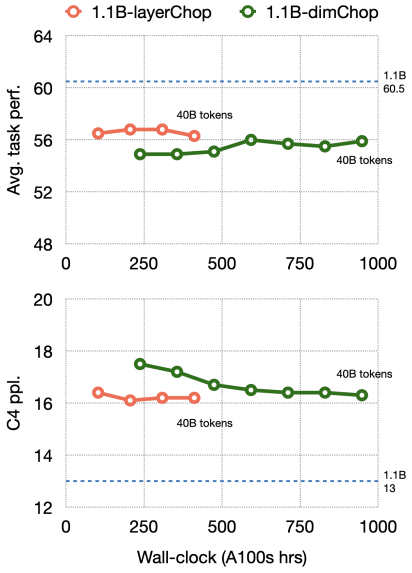

Figure 3: Average task accuracy and language modeling comparison between LayerCHOP and DimCHOP on a fixed compute budget. We demonstrate that LayerCHOP is 2× more efficient during the compression process than DimCHOP by pruning all layers immediately (Appendix C), while DimCHOP opts for a more iterative pruning approach.

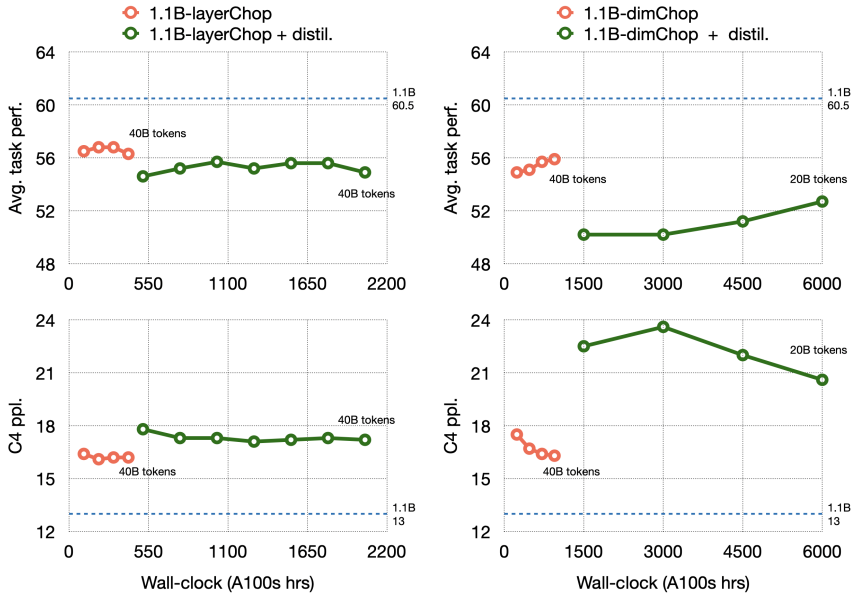

Figure 4: Average task accuracy and language modeling comparison between LayerCHOP, DimCHOP and their distillation loss augmented versions. In a large data regime, for large student and teacher models, we see that adding distillation become inefficient with our pretrain-then-finetune paradigm for LLM compression.

tion 3.3. Homotopic distillation (Liang et al., 2023) finds this setup to be efficient and highly performant for task-agnostic compression of BERT-style models; however, in our large data regime with bigger decoder-only LLMs, we find that augmenting pruning with distillation losses not only underperforms on average task score, when evaluated in a zero-shot manner but also on language modeling.

With a larger teacher and student model, augmenting a pruning setup distillation losses in different parts of the architecture incurs a heavy efficiency penalty, as is evident from Figure 4. Previous distillation approaches are also orthogonal to newer efficient modeling paradigms like flash-attention, which does not materialize a $O(n^2)$ attention matrix to which we can apply the earlier proposed distillation objective in Section 3.3.

6 Discussion and Conclusion

This work discusses how existing LLM compression methods do not follow the pretrain-then-finetune paradigm. We show that with additional continued pretraining of compressed models after pruning, coupled with embarrassingly simple pruning techniques, can beat complex structured pruning approaches and compete with semi-structured compression methods with improved efficiency. Our LayerCHOP methods set new performance standards for LLM compression while improving inference speeds by 1.84× over the baseline model, compared to the limited 1.24× end-to-end speed improvement from 2:4 semi-structured compression via specialized NVIDIA accelerator kernels.

The work closest to ours is Sheared Llama (Xia et al., 2023), where they apply Co-Fi (Xia et al., 2022) like fine-grained pruning and continued pretraining on RedPajama data. Their work selects components of a 7B model (e.g., attention heads) to assemble smaller models, at the 1.3B and 2.7B scales, using a pruning heuristic that trains further on domain-specific data. Our work differs by recovering state-of-the-art compression results by continuing to pretrain despite pruning models at a much coarser granularity. Our methods also require no domain-specific data selection, as we continue pretraining on same pretraining data distribution.

Finally, we demonstrate that under this new pretrain-then-finetune paradigm for LLM compression, augmenting pruning with distillation losses does not improve performance as is the case for BERT-style models. We also identify how student-teacher distillation on large pretraining corpus results in inefficient compression algorithms. CHOP offers compression compatible with contemporary modelling optimizations (e.g., Flash Attention) to surpass distillation with embarassingly simple techniques. We hope our work contributes to a new conversation on how to train a compressed large language model, and how to learn from large corpora for better, smaller models.

Limitations

While our work is in the spirit of reducing model size and improving efficiency — we require significant computational resources for our experiments demanding both high energy usage and processing power. Experiments such as the model pretraining and distillation at scale demand multiple days of training time using 32xA100 GPUs with a high bandwidth interconnect. Therefore, reproducing our experiments are only reasonably tractable with commensurate GPU resources which may be infeasible for some researchers.

Additionally, we demonstrate our findings compared to pretraining and distillation approaches and recently published alternatives in our decoder-only setup. We take this approach to report how the most typical compression strategy can be ported to a contemporary LLM. Our finidings do not indiciate distillation to be a potentially efficient compression strategy for GPT-style models in a large data regime, however, our work is limited in that there may exist *some atypical* distillation strategy with even better performance. We encourage future work and discussion of how these methods can be improved in this regard.

Ethics Statement

We report all pretraining experiments with the widely used C4 corpus. This corpus has been found to contain harmful artifacts and biases (Dodge et al., 2021) which our models may inherit, however, the study of this phenomena is outside of the scope of our work but may inform future study. Model compression has been linked to increased bias and toxicity in a model (Hooker et al., 2020) but it is currently unclear how such effects extend to our setting; particularly as we expose the student to the same corpus as the teacher. Further study is needed in this area to examine how compression influences biases in increasingly large language models (Solaiman et al., 2023).

References

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *ArXiv*, abs/2401.15024.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020a. The lottery ticket hypothesis for pre-trained bert networks. *ArXiv*, abs/2007.12223.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. Symbolic discovery of optimization algorithms. *ArXiv*, abs/2302.06675.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2020b. Early-bert: Efficient bert training via early-bird lottery tickets. In *Annual Meeting of the Association for Computational Linguistics*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim M. Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Paveti'c, Dustin Tran, Thomas Kipf, Mario Luvci'c, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. 2023. Scaling vision transformers to 22 billion parameters. *ArXiv*, abs/2302.05442.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332. Curran Associates, Inc.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *ArXiv*, abs/1909.11556.

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

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

Sanchit Gandhi, Patrick von Platen, and Alexander M. Rush. 2023. Distil-whisper: Robust knowledge distillation via large-scale pseudo labelling. *ArXiv*, abs/2311.00430.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising bias in compressed models.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *ArXiv*, abs/2004.04037.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. In *Findings*.

Sheng-Chun Kao, Amir Yazdanbakhsh, Suvinay Subramanian, Shivani Agrawal, Utku Evci, and Tushar Krishna. 2022. Training recipe for n: M structured sparsity with decaying pruning mask. *ArXiv*, abs/2209.07617.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: A simple depth pruning for large language models. *ArXiv*, abs/2402.02834.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. *ArXiv*, abs/1606.07947.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021a. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021b. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Chen Liang, Haoming Jiang, Zheng Li, Xianfeng Tang, Bin Yin, and Tuo Zhao. 2023. Homodistil: Homotopic task-agnostic distillation of pre-trained transformers. *ArXiv*, abs/2302.09632.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101.

J. Scott McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structured pruning of a bert-based question answering model. *arXiv: Computation and Language*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Neural Information Processing Systems*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Subhabrata Mukherjee, Ahmed Hassan Awadallah, and Jianfeng Gao. 2021. Xtremedistiltransformers: Task transfer for task-agnostic distillation. *ArXiv*, abs/2106.04563.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *ArXiv*, abs/2004.03844.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *ArXiv*, abs/1907.10641.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *ArXiv*, abs/2005.07683.

Noam M. Shazeer. 2020. Glu variants improve transformer. *ArXiv*, abs/2002.05202.

Irene Solaiman, Zeerak Talat, William Agnew, Lama Ahmad, Dylan Baker, Su Lin Blodgett, Hal Daumé III au2, Jesse Dodge, Ellie Evans, Sara Hooker, Yacine Jernite, Alexandra Sasha Luccioni, Alberto Lusoli, Margaret Mitchell, Jessica Newman, Marie-Therese Png, Andrew Strait, and Apostol Vassilev. 2023. Evaluating the social impact of generative ai systems in systems and society.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Raghavi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, A. Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Daniel Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hanna Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. *ArXiv*, abs/2402.00159.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models. *ArXiv*, abs/2306.11695.

S. Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In *Conference on Empirical Methods in Natural Language Processing*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *ArXiv*, abs/2004.02984.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy J. Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *ArXiv*, abs/1903.12136.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. 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, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv: Computation and Language*.

Tycho F. A. van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki Markus Asano, and Tijmen Blankevoort. 2023. The llm surgeon. *ArXiv*, abs/2312.17244.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Elena Voita, David Talbot, F. Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *ArXiv*, abs/1905.09418.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2020a. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020b. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *ArXiv*, abs/2002.10957.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *ArXiv*, abs/1707.06209.

M. Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Annual Meeting of the Association for Computational Linguistics*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *ArXiv*, abs/2310.06694.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*.

Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. PLATON: Pruning large transformer models with upper confidence bound of weight importance. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26809–26823. PMLR.

# Params	Dim	Heads	Layers	Batch Size	LR	Token Budget
180M	1024	16	**12**	2M	6.0e-4	160B
300M	1024	16	24	2M	3.0e-4	160B
610M	2048	16	**12**	2M	2.5e-4	160B
1.1B	2048	16	24	2M	2.0e-4	160B
3.5B	4096	32	**16**	4M	3.0e-4	160B
7B	4086	32	32	4M	3.0e-4	2T

Table 4: Configurations for models used in our experiments at different scale. The 7B model used is pretrained llama2-7B.

A Model Architecture

We list details about our decoder-only models in Table. 4. For the 7B size, we prune a pretrained llama2-7B (Touvron et al., 2023). For smaller model sizes, we train a custom architecture model on 160B C4 tokens.

For the custom model, we follow the PaLM architecture (Chowdhery et al., 2022) owing to improved throughput efficiency. Specifically, the attention and feed-forward network (FFN) modules are parallel instead of sequential (Radford et al., 2019). SwiGLU activation (Shazeer, 2020) is used in the FFN module. Multi-head attention uses the equivalent Flash-Attention (Dao et al., 2022, FA) implementation. The first layer of the FFN module and the layers generating attention query, key, and value are fused. Similarly, the second layer of the FFN module and the feed-forward layer after the attention operation are fused. The LayerNorm (Ba et al., 2016) is before the first fused feed-forward layer. The query and the key vectors are passed through additional layer normalization layers for increased training stability following Dehghani et al. (2023). This block structure is repeated with skip connections to form our decoder-only Transformer architecture.

B Evaluation setup

We detail the tasks used in our zero-shot evaluation suite in Table 5. Each task in the table reports either classification accuracy or length normalized classification accuracy. Our evaluation suite, which is online (runs as a validation loop after some training steps), is adapted to match the results from Eleuther AI eval harness (Gao et al., 2021).

C LayerCHOP

Our baseline models contain 24 decoder layers. To determine which layers we should prune for the best compression performance, we define five

category	task	metric
common sense reasoning	PIQA (Bisk et al., 2019)	len norm acc
	Hellaswag (Zellers et al., 2019)	len norm acc
	Winogrande (Sakaguchi et al., 2019)	acc
science question answering	OpenBookQA (Mihaylov et al., 2018)	len norm acc
	SciQ (Welbl et al., 2017)	acc
	Arc-Easy (Clark et al., 2018)	acc

Table 5: Zero-shot downstream tasks for evaluating our compressed models and baselines. Each task either reports classification accuracy or length normalized classification accuracy.

Model	Token Budget	Task Metric	max-gap	input	output	middle	both	Pre-compression
300M-160B	20B	ppl (↓)	**23.0**	24.5	25.6	24.0	**23.0**	16.2
300M-160B	20B	avg acc (↑)	**53.2**	52.9	51.6	52.9	**53.2**	55.8
1.1B-160B	20B	ppl (↓)	18.1	**17.3**	22.0	18.6	18.6	13.0
1.1B-160B	20B	avg acc (↑)	54.8	**55.1**	53.1	54.7	53.8	59.8

Table 6: Influence on average task performance and C4 validation perplexity of different truncated initialization strategies from Figure 5 for models of size 300M and 1.1B. The average task performance score is across 6 tasks listed in Table 5, and higher numbers are better. For perplexity scores, lower is better.

layer pruning configurations shown in Figure 5, each removing 12 out of the 24 decoder layers of the 300M and 1.1B models. In all these pruning configurations, we always keep the first and the last layers because they interact with the embedding table. We made this design choice based on early experiments. Table 6 summarizes the results of this ablation. We report the perplexity score on the C4 validation set and the average task accuracy across 6 tasks in our evaluation suite (Table 5).

For the base 300M model, pruning configurations of *max-gap* and *both* perform the best out of the five possible configurations. For the 1.1B model, pruning layers from the *input* configuration yielded the best results for both reported metrics. The *output* pruning configuration resulted in the worst performance across model sizes, suggesting that pruning layers towards the output side of the model should be avoided. Given these results, we use the pruning configuration of *max-gap* for all our 300M model experiments and the configuration of *input* for all our 1.1B model experiments, and use the configuration from 1.1B model experiments for our llama2-7B pruning experiments.

To answer the question of when we should prune layers from our model while continuing to train it with language modeling loss, we can decide in one of two ways: either remove selected layers at once or remove them one by one, each after a fixed number of training tokens. We run this experiment in four configurations to see if increasing the gap between each layer pruning increases training stability or model performance. The four configurations are: dropping all layers at once (0M token gap between pruning each layer) or pruning them after 100M, 500M, and 1B training tokens each. We run this experiment for the 300M and 1.1B model sizes. We prune 12/24 layers from our decoder-only models for this ablation, each at a training token gap mentioned in one of the four configurations above. The result of this experiment is summarized in Figure 6. Pruning layers one by one with an increasing token budget between each layer pruning does not benefit the average task accuracy or C4 validation perplexity. In fact, there is a marginal preference to prune layers as early into the training as possible. Hence, we decided to prune all 12/24 layers

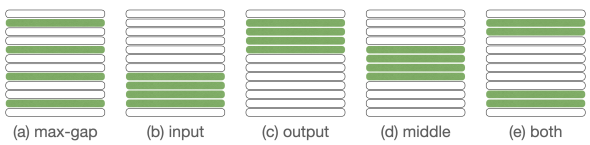

Figure 5: Truncated initialization configurations for layer pruning in a decoder-only language model. Highlighted layers (green) are removed. We retain the first and last layer as these layers interact with the embedding table.

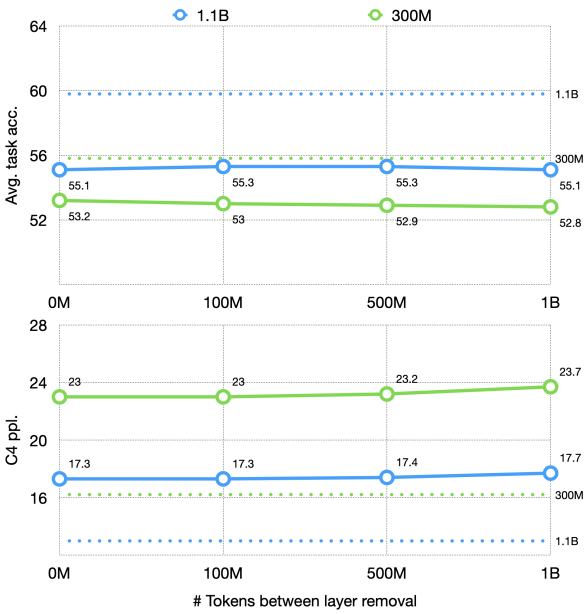

Figure 6: Average task accuracy (Table 5) and perplexity on the C4 validation set for model sizes 300M and 1.1B comparing schedules for *when* to prune layers during continued pretraining. We find a marginal performance degradation as we remove layers one by one further apart during continued pretraining.

simultaneously for other experiments.

D DimCHOP