Large Language Models Are Overparameterized Text Encoders

Anonymous ACL submission

Abstract

Large language models (LLMs) demonstrate strong performance as text embedding models when finetuned with supervised contrastive training. However, their large size balloons inference time and memory requirements. In this paper, we show that by pruning the last $p\%$ layers of an LLM before supervised training for only 1000 steps, we can achieve a proportional reduction in memory and inference time. We evaluate four different state-of-the-art LLMs on text embedding tasks and find that our method can prune up to 30% of layers with negligible impact on performance and up to 80% with only a modest drop. With only three lines of code, our method is easily implemented in any pipeline for transforming LLMs to text encoders. We also propose L^3Prune, a novel layer-pruning strategy based on the model's initial loss that provides two optimal pruning configurations: a large variant with negligible performance loss and a small variant for resource-constrained settings. On average, the large variant prunes 21% of the parameters with a -0.3 performance drop, and the small variant only suffers from a -5.1 decrease while pruning 74% of the model. We consider these results strong evidence that LLMs are overparameterized for text embedding tasks, and can be easily pruned.

1 Introduction

In the past few years, the field of natural language processing (NLP) has seen a significant shift towards large-scale language models (LLMs). These models, due to a combination of their large size, extensive pre-training, and instruction-following ability, have achieved state-of-the-art performance on a wide range of NLP tasks, such as language modeling, text generation, and text understanding (Dubey et al., 2024; Brown et al., 2020; Jiang et al., 2023a).

Despite their strong generative capabilities, decoder-only LLMs have seen comparatively little adoption for text embedding tasks until recently (BehnamGhader et al., 2024). Text embedding, which involves mapping a text sequence of varying length to a fixed-dimensional vector representation, is a fundamental task in NLP and is used as a building block for a wide range of downstream tasks, such as semantic textual similarity, information retrieval, and text classification. Further, it is a fundamental step required for retrieval-augmented generation, a well-known paradigm for improving the performance of LLMs in knowledge-intensive tasks (Lewis et al., 2020). Traditionally, text embedding models have been based on masked language models (MLMs) and bidirectional encoders, such as BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020), typically adapted for text embedding tasks by following a multi-step training pipeline consisting of weakly- and fully-supervised contrastive training (Ni et al., 2022; Li et al., 2023; Xiao et al., 2024).

Decoder-only LLMs, however, offer several advantages over their encoder-only counterparts. They are more sample-efficient during pre-training, leverage instruction-following capabilities for task generalization, and benefit from a rich and evolving research ecosystem (Clark et al., 2020; Asai et al., 2023; BehnamGhader et al., 2024). Further, the availability of high-performing public pre-trained LLMs and their continual development make it appealing to explore their use for text embedding tasks. To this end, several studies have experimented with various pipelines, training methods, and architectural modifications, effectively converting LLMs into state-of-the-art text embedding models with small amounts of supervised contrastive training (BehnamGhader et al., 2024; Li and Li, 2024; Ma et al., 2024; Muennighoff, 2022; Springer et al., 2024; Lee et al., 2024).

On the other hand, the increasingly large size of LLMs, with parameters ranging up to 540B (Brown et al., 2020; Chowdhery et al., 2023; Dubey et al.,

2024), stands in stark contrast to traditional small bidirectional encoders of sizes almost universally less than 1B parameters (Li et al., 2023; Xiao et al., 2024). Even the smallest LLMs in use typically have 3-8B parameters (Abdin et al., 2024). Consequently, inference with LLM-based text encoders is far more demanding in terms of compute and memory requirements in comparison to traditional methods.

Therefore, there are a variety of post-training techniques for reducing the cost of LLMs, such as pruning, quantization, and distillation (Zhu et al., 2024). In particular, the recent work of Gromov et al. (2024) has shown that LLMs can be pruned to up to half their size with minimal impact on downstream performance (i.e. question answering) by dropping the last half of the model's layers, with the exception of the final layer, and applying a small amount of parameter-efficient finetuning. Layer-dropping as a pruning strategy has particular benefits: it is straightforward to implement, with memory and inference time decreasing linearly with the number of layers dropped, and it can be combined with other efficiency methods such as quantization.

In this work, we build on these findings and apply them in the context of text embedding, resulting in a easy-to-use and efficient approach to transform any pre-trained decoder-only LLM into a much smaller text embedding model. By simply pruning the last $n\%$ layers of a model before supervised contrastive training, we reduce the final model size with a proportional decrease in memory and inference time. We experiment with four different decoder-only LLMs ranging from 3.8B to 8B parameters with a variety of pruning percentages and show that up to 30% of a model's layers may be pruned with almost no impact in performance, and may even *increase* it. Even intensive pruning on the order of 80% still provide reasonably effective text embedding models, with a drop in performance on the downstream task from 64.9 to 59.8 for our highest-performing model.

Further, we propose **L^{3}Prune**, a simple and novel method that pinpoints particular layers to prune to based on the initial without requiring significant testing or experimentation. With no input, our method produces both *a)* a lightly-pruned model, 69-89% of the original size with minimal performance loss of -0.2 on average and even a performance *improvement* in one model, and *b)* a heavily pruned model, 16-36% of their original size with a modest performance drop of -4.4 to -6.9.

We note that our pruning technique is both orthogonal to other commonly used pruning and quantization methods and to the specifics of the methodology used to finetune the LLM for text embedding. As such, it can be seamlessly integrated with other efficiency techniques and finetuning methods to further optimize model size and performance.

Our contributions can be summarized as follows:

- We are the first to apply pruning in a text embedding setting, formulating a simple procedure that can be easily applied to pipelines converting an LLM to a text encoder.
- We demonstrate that LLMs can be pruned by up to 30% with negligible impact on the quality of representations and up to 80% with a modest performance drop.
- We propose and evaluate L^{3}Prune, a novel method that identifies layers to prune by leveraging the model's initial loss, thus minimizing the need for trial-and-error for effective pruning.

Overall, our work demonstrates that decoder-only LLMs are generally overparameterized for text embedding tasks, and that significant reductions in model size can be achieved with minimal impact on performance. We release the full code for L^{3}Prune [1].

2 Related Work

2.1 Encoder-only Text Embedding Models

BERT-based models have largely dominated the field of text representation in the past, relying on supervised training with natural language inference or sentence similarity to produce high-quality sentence embeddings (Conneau et al., 2017; Reimers and Gurevych, 2019). Recent methods have further improved these representations through large-scale contrastive pretraining followed by multi-task finetuning (Ni et al., 2022; Wang et al., 2022; Li et al., 2023; Xiao et al., 2024). These methods generally require a complex multi-stage training pipeline that demands substantial engineering effort, along with large-scale compute-intensive pretraining (Zhang et al., 2024).

[1] https://anonymous.4open.science/r/l3prune-1072/

2.2 Decoder-only Text Embedding Models

A variety of recent works have explored leveraging LLMs and their capabilities to generate high-quality text representations. Generally, a combination of *(a)* a pooling method, *(b)* architectural modifications, and *(c)* supervised or unsupervised fine-tuning is used to effectively convert LLMs to text embedding models.

The majority of prior work consider two straightforward pooling strategies to extract embeddings for a sequence of tokens: mean pooling and last-token pooling (Springer et al., 2024; Jiang et al., 2023b; BehnamGhader et al., 2024; Muennighoff, 2022; Wang et al., 2024). Mean pooling is more effective with bidirectional embedding models (BehnamGhader et al., 2024; Wang et al., 2022) while last-token pooling is generally preferred when working with causal attention (Lee et al., 2024; BehnamGhader et al., 2024). Muennighoff (2022) introduces weighted mean pooling, assigning a higher weight to later tokens to offset the autoregressive nature of decoder-only LLMs, with significant success. Lee et al. (2024) utilizes a trainable latent attention layer as a pooling technique and obtains consistent improvement.

Several studies identify the causal attention mechanism of decoder-only LLMs as a obstacle in obtaining performant representations, and suggest modifications to the architecture to compensate. Li and Li (2024) and BehnamGhader et al. (2024) replace the causal attention mechanism with bidirectional attention. Muennighoff et al. (2024) utilizes a hybrid objective with both bidirectional representation learning and causal generative training. Lee et al. (2024) finds that simply removing the causal attention mask works compellingly well.

Finally, both supervised and unsupervised fine-tuning have been extensively explored to significantly improve the performance of decoder-only LLMs in representational tasks, with supervised training consistently producing the best results (BehnamGhader et al., 2024; Muennighoff, 2022; Jiang et al., 2023b). Several modifications to the training pipeline have been proposed, such as an additional masked token prediction training step (BehnamGhader et al., 2024), or a two-stage instruction-tuning setup (Lee et al., 2024). The zero-shot setting has also been studied with limited success by Springer et al. (2024) and Jiang et al. (2023b).

2.3 LLM Pruning

Pruning as a method of size reduction has a long history in the field of deep learning (Cheng et al., 2024). Classic pruning techniques sparsify networks by removing individual parameters based on various criteria (LeCun et al., 1990; Han et al., 2015). While these models were smaller, these techniques generally lead to irregular sparsification patterns that require specialized hardware or libraries to fully utilize. Structured pruning techniques were developed to remove irrelevant groups of parameters together, such as particular channels or filters in convolutional neural networks (Wen et al., 2016; Li et al., 2022).

Recent work has focused on applying structure pruning methods to transformers. Almost every possible component of the model architecture is studied as candidates for removal, most prominently methods that drop attention heads (Voita et al., 2019; Michel et al., 2019; Kim and Hassan, 2020) and layers (Fan et al., 2020; Zhang et al., 2022; Sajjad et al., 2023; Gromov et al., 2024; Men et al., 2024; Fan et al., 2024). Prior literature on layer pruning generally consider BERT-like models (Fan et al., 2020; Sajjad et al., 2023), with recent studies shifting focus to decoder-only LLMs (Gromov et al., 2024; Men et al., 2024; Fan et al., 2024).

Sajjad et al. (2023) finds that for BERT-like models, dropping the last layers is the best layer pruning strategy. Gromov et al. (2024) extends this research to decoder-only LLMs, and presents a layer pruning strategy, pruning a block of layers based on angular distance between layer representations. Their results indicate that the last layer in particular is essential for maintaining performance. Informed by this finding, they propose a simpler strategy: dropping the last n layers with the exception of the final layer. They conclude that simply dropping the last layers works effectively to prune the model, with a caveat: after dropping the layers, it is required to "heal" the model via finetuning with QLoRA (Dettmers et al., 2023) for 1000 steps.

While these results suggest that the last layer in particular is essential when pruning LLMs for text generation, this is not necessarily the case when utilizing the LLM for other tasks. To this end, Fan et al. (2024) finds that for "simpler" tasks such as sentiment analysis, early stopping—stopping the inference after a certain number of layers—is an effective strategy to significantly reduce inference time with minimal impact on performance. The

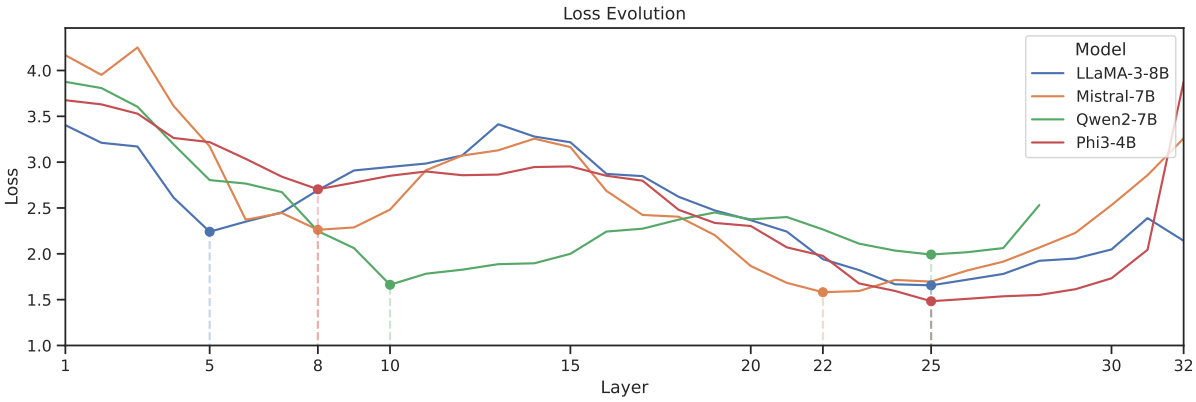

Figure 1: The loss values extracted from the layerwise embeddings of samples from the training dataset. These values were obtained for each unmodified model. The marked points indicate the layer with minimal loss before and after the midpoint.

authors suggest that the later layers of LLMs, including the final layer, may not be necessary when using the LLM representations for other tasks.

3 Pruning

We borrow the intuition from Gromov et al. (2024), that the representations in a transformer can be thought of as a slowly changing function of the layer index. Specifically, the representation can be formulated as the following iterative residual equation:

$$x^{(\ell+1)} = x^{(\ell)} + f(x^{(\ell)}, \theta^{(\ell)}), \tag{1}$$

where $x^{(\ell)}, \theta^{(\ell)}$, respectively, are the multi-dimensional input and parameter vectors for layer ℓ, and $f(x, \theta)$ describes the transformation of one multi-head self-attention and MLP layer block.

The authors assert that these representations converge to a slowly changing function:

$$x^{(\ell)} \approx x^{(\ell-1)} + \epsilon \tag{2}$$

with $\epsilon \ll x^{\ell}$ as an approximation. They verify this hypothesis experimentally by calculating the distance between layer representations and using them for a pruning algorithm. Their findings indicate that the earlier layers have a significantly larger impact on the representation compared to the later layers, with a particular caveat: the final layer also modifies the representation significantly. They thus propose and verify a simpler pruning strategy, where the last n layers of the model, excluding the final layer, are dropped. This method non-optionally requires a "healing" step, recovering the downstream performance with a small number of QLoRA finetuning steps (Dettmers et al., 2023).

Our hypothesis extends theirs, and posits that for the task of generating text embeddings, the final layer is also not necessary. Our pruning experiments are conducted with the percentage pruned p, between 0% (all layers intact) and 100% (all layers removed). Given a pruning percentage and a total number of layers n, the new number of layers n^* is calculated as

$$n^* = \lfloor n \times (1 - p) \rfloor$$

Given a model and its configuration, this straightforward procedure can be integrated with modern LLM implementations with just three lines of code:

```
n = int(config.num_hidden_layers * (1-p))
model.layers = model.layers[:n]
config.num_hidden_layers = n
```

We then conduct supervised contrastive training, as with prior work on converting LLMs to text encoders. In lieu of an explicit healing step, we hypothesize that the aforementioned training acts as such. Thus, no additional or separate training is necessary to execute our method.

4 Experiments

4.1 General Setup

For our experiments, we chose four instruct-tuned decoder-only LLMs across different families ranging from 3.7B to 7.5B: LLaMA-3-8B (`Meta-Llama-3-8B-Instruct`, Dubey et al., 2024), Mistral-7B (`Mistral-7B-Instruct-v0.2`, Jiang et al., 2023a), Qwen2-7B (`Qwen2-7B-Instruct`, Yang et al., 2024), and Phi3-4B (`Phi-3-mini-4k-instruct`, Abdin et al., 2024). These model families were chosen

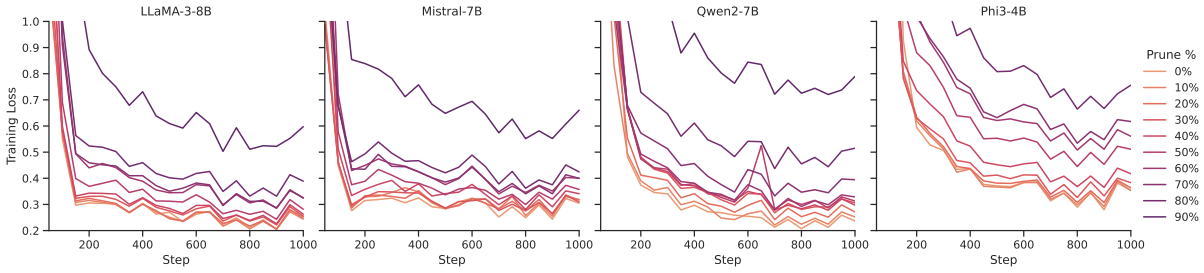

Figure 2: The training loss curves for each model at different pruning percentages.

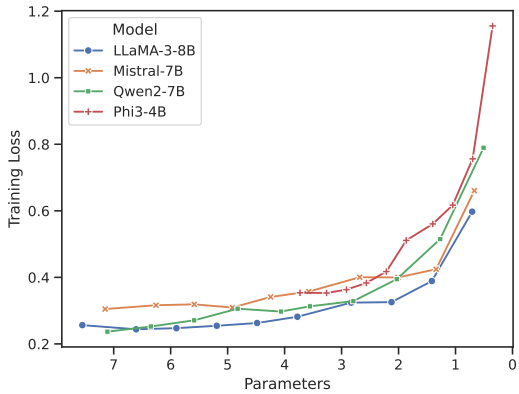

Figure 3: The final loss values at the end of training across different pruning percentages.

due to their widespread use in open-source communities and LLM literature. As we are chiefly concerned with the effects of pruning, we opt for no modification to the LLM architecture itself. We use weighted mean pooling (Muennighoff, 2022) to generate embeddings from the outputs of the LLM as it is straightforward to implement and outperforms other pooling measures when paired with causal attention (Muennighoff, 2022; BehnamGhader et al., 2024).

We also conduct supervised contrastive finetuning, known to outperform unsupervised finetuning and the zero-shot setting, and considered to be an integral part of effectively utilizing LLMs as embedding models (BehnamGhader et al., 2024; Muennighoff, 2022; Jiang et al., 2023b). We use the replication of the public portion of the E5 dataset (Wang et al., 2024), curated by Springer et al. (2024), as the training dataset. Consisting of approximately 1.5 million samples, it is a multilingual compilation of various retrieval datasets, meant for supervised contrastive training of embedding models. In accordance, we use contrastive loss with hard negatives and in-batch negatives (Springer et al., 2024; BehnamGhader et al., 2024). Further details on the dataset and training are provided in Appendix A.

All experiments were conducted on a single A100 GPU, reinforcing the accessible nature of our proposed procedure.

4.2 Zero-shot Loss Evolution Over Layers

As a preliminary test of our hypothesis—that an LLM can form performant text representations even before reaching the final layer—we first calculate how well the output of each layer of the model performs as an embedding. We note that this is equivalent to a zero-shot setting. As we are interested in a comparative measure between layers intra-model, the loss as a metric is sufficient. We take a random sampling of 1280 tuples from the training dataset and calculate the embeddings via weighted mean pooling of the outputs of each layer. Then, the loss is calculated and averaged per layer. We find that the loss values converge fairly quickly, and so 1280 samples are sufficient for our purposes. The results are aggregated in Figure 1.

The loss for all four models follow a similar curve: an initial drop to around layer 5-10, a subsequent rise around layer 15, and then a slower drop up to layer 22-25, where it rises again by the end with layer 28-32. While the specifics of the ways in which LLM representations evolve are not well-understood, these results suggest that the early layers of the model are generally focused on representation, while the final layers transform the representation into the specific probability distribution for the next token. Regardless of the underlying dynamics, the drop-rise-drop curve is consistent across model sizes and families in our experiments.

We expect that training will transform the shape of this layerwise evolution considerably. We also have little reason to expect that the final downstream performance of layer-dropped models will be accurately modeled by the effectiveness of these initial representations. However, we posit that

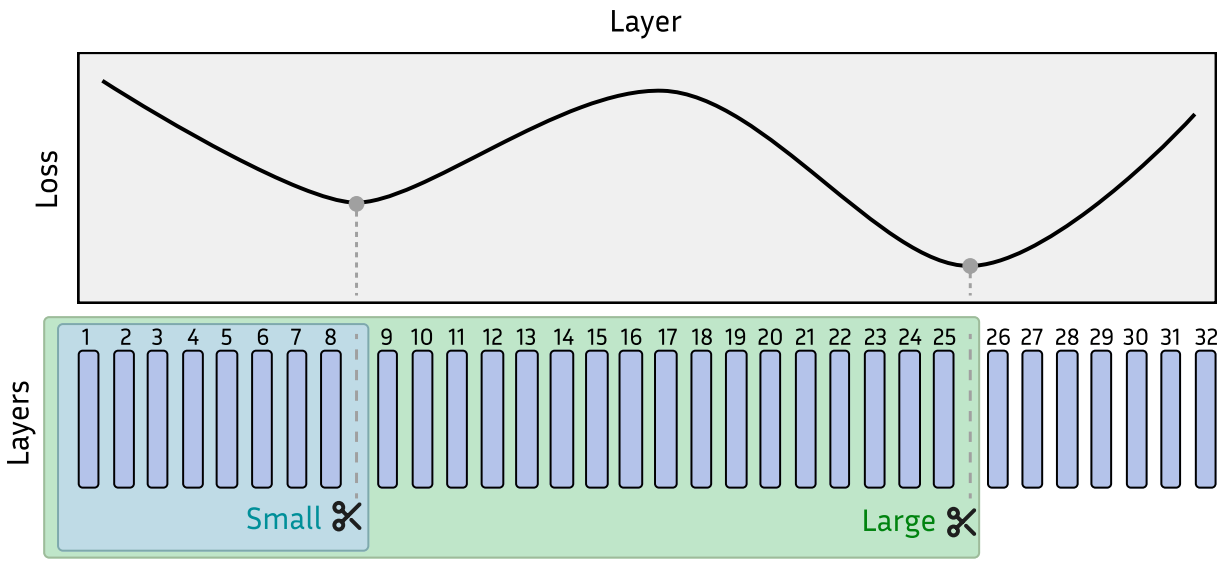

Figure 4: A simplified illustration of L^{3}Prune. The initial loss of the representation of each layer is found, and the two minima before and after 50% of the model correspond to the layers to prune to in the two configurations, `small` and `large`.

these initial loss curves also reveal optimal starting points for pruning. The minima of these curves indicate layers where the text embeddings are best-optimized, making them good candidates for pruning without significant performance loss.

Inspired by these findings, we consider the following procedure for pruning: find the two minima in the layer-loss curve before and after 50% of the layers (the low point of the two drops). We hypothesize that pruning up to these layers provides us with two models: a smaller model with degraded but reasonable performance, and a larger model whose performance is close to the original. This procedure would thus produce two text embedding model variants from an LLM, each variant usable in different circumstances. The two aforementioned models are termed `large` and `small` in the following sections. We term this method **L**LM **L**ayerwise **L**oss **P**runing, or **L^{3}Prune** for short. Figure 4 shows a simple illustration of the process.

4.3 Supervised Training

In order to verify the general efficacy of our hypothesis—that LLMs can form effective text representations even before reaching their deeper layers—we conduct training on pruned LLMs to convert them into effective text encoders. We keep the training procedure fairly straightforward: supervised contrastive learning for 1000 steps with LoRA modules (Hu et al., 2022). Other hyperparameters are detailed in Appendix A.2. We first test a range of pruning percentages from 10% to 90% at 10% intervals. Figure 2 shows the training loss for all models and pruning percentages. We note that the training loss curves all generally follow the same shape, indicating stability in training even with the modified architecture.

Figure 3 shows the final loss in relation to the pruned model parameters, with each marked point representing a model pruned by an additional 10%. The final loss values for each model follow a straightforward trajectory with increasing pruning percentage: minimal increases up to 30-40%, with larger increases as the pruning percentage hits 90%. Notably, we find that the final loss of different models correlates more with the final parameter count after pruning than with the percentage of layers retained. This suggests that the parameter count is a more significant factor in determining the effectiveness of a pruned model than simply the proportion of layers kept.

If we presume that training loss correlates well with downstream accuracy for text embedding, we can make a series of predictions from an analysis of the plots:

- Performance always degrades sharply as the parameter count approaches and goes below 1 billion.

- In contrast, performance degrades little even with 30-50% pruning. LLaMA-3-8B degrades minimally up to 40-50%, Mistral-7B up to 30-40%, and Phi3-4B up to 20-30%. Qwen2-7B

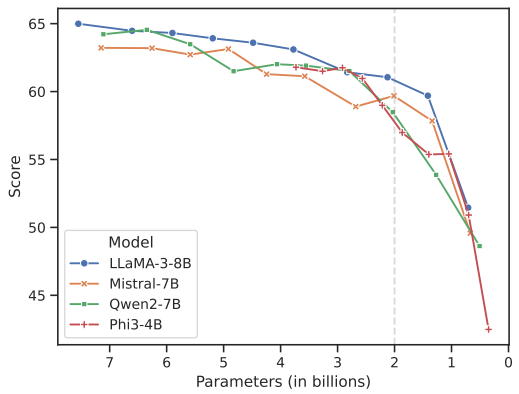

Figure 5: The MTEB (15 task subset) scores with respect to the number of model parameters.

degrades more at low pruning percentages, but remains stable between 30-60%.

- Even at high pruning percentages, model performance degrades at a reasonable rate. Models can likely be pruned up to 2 billion parameters while still producing viable embeddings.

4.4 Simple Pruning Evaluation

To validate the predictions made from the training loss, we evaluate the models at various pruning percentages on downstream text embedding tasks. Specifically, to speed up evaluation, we opt for the 15-task subset of the Massive Text Embedding Benchmark (MTEB, Muennighoff et al., 2023) collected and used by BehnamGhader et al. (2024). The subset, which we term MTEB-15 for clarity, covers representative tasks from the full 56 tasks in MTEB, including tasks from each category with almost the same proportion to prevent bias. Further details are provided in Appendix B.1.

In accordance with previous work (BehnamGhader et al., 2024; Springer et al., 2024; Wang et al., 2024), we evaluate with task-specific instructions. We use the same instructions as Wang et al. (2024). The instructions can be found in Appendix Table 4. Following BehnamGhader et al. (2024), for symmetric tasks, the same instruction is used for the query and the document. Instruction tokens are excluded from the final pooling.

Figure 5 shows the impact of pruning on MTEB-15 results across a range of pruning percentages. We plot with respect to the number of parameters as opposed to relative pruning percentages because parameter count correlates better with the score. We can see that the training loss and MTEB-15 score also roughly correlate. This confirms that our predictions in Section 4.3, based on the supervised training loss, are fairly accurate.

LLama at 50% pruning (3.77B) is only degraded by -1.89, still providing a strong performance of 63.10. Even at 80% pruning (1.41B), it performs at a reasonable 59.69. Mistral's performance decrease is an almost negligible -0.08 up to 30% (4.91B). Qwen's performance *increases* by +0.32 with a pruning of 10%. It drops distinctly at 30% pruning. However, it stabilizes at a reasonable 61.51 up till 60% (2.79B). Phi degrades negligibly up to 20% pruning (2.91B) with -0.03, and -0.53 at 30% (2.56B). Higher pruning percentages degrade it significantly, as the model parameter count decreases below the 2 billion mark.

Our results correspond roughly with those of Gromov et al. (2024): sharp transitions in performance around 45%-55% for models in the Llama family, 35% for Mistral, 25% for Phi, and 20% for Qwen. However, instead of a sharp transition to near-random performance, we observe a steady but reasonable decline even at higher pruning percentages. In general, we only observe a significant decline in performance as model size goes below roughly 2 billion parameters. These results also correlate roughly with previous findings by Jiang et al. (2023b), who investigated LLM-based sentence embedding models between 125M to 66B parameters and found diminishing returns at parameter counts over 2B.

We can derive some general insights from these experiments. For one, the resilience of a model to pruning is not entirely consistent across families and sizes. Thus, model-specific experimentation may be required. However, in general, models can be pruned 10-30% with minimal drop in downstream performance. Further, higher pruning percentages up to 80% still yield reasonably effective embedding models.

We note that LLaMa-3-8B at 50% pruning, with 3.77B parameters, outperforms an unpruned Phi3-4B at 3.73B parameters. In conjunction with our other results, we suggest that given a compute/memory budget, simply dropping layers of a high-performing LLM may be a superior and significantly simpler strategy than training a smaller LM that fits the budget.

	LLaMA-3-8B		Mistral-7B		Qwen2-7B		Phi3-4B	
	Large	**Small**	**Large**	**Small**	**Large**	**Small**	**Large**	**Small**
Layers	25 (-7)	5 (-27)	22 (-10)	8 (-24)	25 (-3)	10 (-18)	25 (-7)	8 (-24)
Params	5.9 (78%)	1.18 (16%)	4.92 (69%)	1.79 (25%)	6.35 (89%)	2.54 (36%)	2.91 (78%)	0.93 (25%)
Score	63.5 (-1.5)	58.1 (-6.9)	63.1 (-0.1)	59.0 (-4.2)	64.5 (+0.3)	60.9 (-3.3)	61.7 (-0.1)	55.5 (-6.3)

Table 1: Comparison of `large` and `small` variants across various models, including number of layers, parameters, and MTEB scores. Changes from the full model are provided in paranthesis.

4.5 L^{3}Prune Evaluation

As mentioned in Section 4.2, we hypothesize that the minima in the layer-loss curve before and after the midpoint are particularly effective points for pruning. We prune to those particular layers and conduct the same training and evaluation as described in Sections 4.3 and 4.4. Table 1 aggregates the results across base models for the two resulting prune configurations, termed `small` and `large`. It also shows the particular layer numbers and parameter counts.

The results are consistent with our previous findings. The `small` models generally perform worse than the full-sized models, with performance drops ranging between -4.4 and -6.9. However, at 16%-36% of their original size (84%-64% pruning), the models are proportionally compute and memory-efficient in exchange for the dropped performance. The `large` models, on the other hand, perform almost as well as the unpruned models, with only a slight drop in performance, while pruned to 69%-89% (31%-11% pruning). As we have seen before, Qwen2-7B's performance *increases* slightly with pruning, and both Mistral-7B and Phi3-4B's performance drops are negligible. LLaMA-3-8B's performance drops by -1.4 points but still remains a fairly strong 63.5.

Combined with the results from Section 4.4, we can see that the layers picked by L^{3}Prune are generally optimal. For instance, Mistral-7B, Qwen2-7B, and Phi3-4B show strong performances up to 30%, 10%, and 20% pruning respectively, and the layers corresponding to those pruning percentages are exactly the layers pinpointed by L^{3}Prune for the `large` variant. As LLaMa-3-8B's performance decrease remains fairly consistent when pruning below 50%, we infer that there is no particularly optimal point for pruning. Similarly, the `small` variants are pruned up to the point before each model's performance drops drastically—roughly 85% for LLAMA-3-8B, 75% for Mistral-7B, 65% for Qwen2-7B, and 75% for Phi3-4B.

Based on these results, we can conclude that the layerwise loss evolution of a model can be used to effectively pick optimal points for pruning. The resulting variants can be used to provide a range of models with different performance and efficiency trade-offs. The `large` models are particularly effective, with a negligible drop (or even an increase) in performance for a significant reduction in size. The `small` models can be used for resource-constrained settings, with reasonable performance.

We further note that the training of the `small` variants required only 23.6 GB of VRAM at maximum, and the layerwise loss curves can be calculated with less than 17 GB of VRAM. The training is only conducted for 1000 steps and takes less than an hour on average using an A100 GPU. Thus, `small` variant models can be trained on consumer-grade GPUs, making it accessible to open-source and practitioner communities. Further details on training times are given in Appendix A.3.

5 Conclusion

In this work, we presented a simple and effective pruning approach to convert LLMs into lightweight, performant text embedding models. By dropping the last $p\%$ layers of the model, we achieved significant reductions in model size and inference time, with minimal impact on text embedding tasks. Our procedure is straightforward to implement in pipelines converting LLMs to text encoders, and requires no additional training, providing smaller models at no cost. Based on the initial model loss, we also proposed L^{3}Prune, a method to pinpoint optimal layers to prune to, providing an efficient strategy for pruning without extensive experimentation. We demonstrated that significant pruning—up to 31%—can be conducted with a negligible performance loss, and substantial pruning—up to 84%—can still produce effective models. Overall, our results show that LLMs are significantly overparameterized for text embedding tasks, and can be pruned with minimal performance loss.

6 Limitations

Our work only takes into consideration the supervised finetuning setting for utilizing LLMs as text encoders, as this is the most common and generally effective. Further, our results may not hold with extensive modifications to the architecture or training process. Lastly, even with extensive pruning, our smallest models are still generally larger than traditionally trained encoder-only models. However, the inexpensive finetuning required and the performance advantages offered by LLM-based text encoders motivates pruning as a viable alternative in resource-constrained settings.

7 Ethical Considerations

Our work provides an effective and efficient method to produce optimized text embedding models from LLMs. As we mentioned in Section 4.5, our method is memory and compute-efficient, and can be conducted on consumer-grade GPUs, making it accessible for a wider audience of practitioners and academics. However, this also enhances potential issues with misuse, lowering the bar for malicious actors to train and host embedding models. Regardless, embedding models in general have significantly fewer avenues for malicious behaviour in comparison to e.g. generative LLMs.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2023. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3650–3675, Toronto, Canada. Association for Computational Linguistics.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. LLM2vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*, Pennsylvania, United States.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115, Louisiana, United States. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages

4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. 2024. Not all layers of llms are necessary during inference. *Preprint*, arXiv:2403.02181.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *Preprint*, arXiv:2403.17887.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28, Montréal, Canada. Curran Associates, Inc.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. DuReader: a Chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 37–46, Melbourne, Australia. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, Online.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023b. Scaling sentence embeddings with large language models. *Preprint*, arXiv:2307.16645.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Young Jin Kim and Hany Hassan. 2020. FastFormers: Highly efficient transformer models for natural language understanding. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 149–158, Online. Association for Computational Linguistics.

Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. 1990. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-Embed: Improved techniques for training LLMs as generalist embedding models. *Preprint*, arXiv:2405.17428.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2022. Pruning filters for efficient ConvNets. In *International Conference on Learning Representations*, Toulon, France.

Xianming Li and Jing Li. 2024. BeLLM: Backward dependency enhanced large language model for sentence embeddings. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 792–804, Mexico City, Mexico. Association for Computational Linguistics.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2421–2425.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *Preprint*, arXiv:2403.03853.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Niklas Muennighoff. 2022. SGPT: GPT sentence embeddings for semantic search. *Preprint*, arXiv:2202.08904.

Niklas Muennighoff, Hongjin SU, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. In *ICLR 2024 Workshop: How Far Are We From AGI*, Vienna, Austria.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech Language*, 77:101429.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2024. Repetition Improves Language Model Embeddings. *Preprint*, arXiv:2402.15449.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, Barcelona, Spain.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 641–649, New York, NY, USA. Association for Computing Machinery.

Xiaohui Xie, Qian Dong, Bingning Wang, Feiyang Lv, Ting Yao, Weinan Gan, Zhijing Wu, Xiangsheng Li, Haitao Li, Yiqun Liu, and Jin Ma. 2023. T2ranking: A large-scale chinese benchmark for passage ranking. In *Proceedings of the 46th International ACM SIGIR*

Conference on Research and Development in Information Retrieval, SIGIR '23, page 2681–2690, New York, NY, United States. Association for Computing Machinery.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. 2021. Mr. TyDi: A multi-lingual benchmark for dense retrieval. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 127–137, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. MIRACL: A Multilingual Retrieval Dataset Covering 18 Diverse Languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131.

Zhehao Zhang, Yan Gao, and Jian-Guang Lou. 2024. e^5: Zero-shot hierarchical table analysis using augmented LLMs via explain, extract, execute, exhibit and extrapolate. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1244–1258, Mexico City, Mexico. Association for Computational Linguistics.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. 2022. PCEE-BERT: Accelerating BERT inference via patient and confident early exiting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338, Seattle, United States. Association for Computational Linguistics.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Preprint*, arXiv:2308.07633.

A Training

A.1 Dataset

The dataset we use consists of ELI5 (sample ratio 0.1, Fan et al., 2019), HotpotQA (Yang et al., 2018), FEVER (Thorne et al., 2018), MIRACL (Zhang et al., 2023), MS-MARCO passage ranking (sample ratio 0.5) and document ranking (sample ratio 0.2, Bajaj et al., 2018), NQ (Karpukhin et al., 2020), SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), Quora Duplicate Questions[2] (sample ratio 0.1), Mr- TyDi (Zhang et al., 2021), DuReader (He et al., 2018), and T2Ranking (sample ratio 0.5, Xie et al., 2023). The instructions used for each dataset can be found in Table 5.

A.2 Hyperparameters

All models are trained with LoRA rank $r = 16$ and use brain floating point (`bfloat16`) precision, gradient checkpointing, and FlashAttention-2 (Dao, 2024) to optimize GPU memory consumption. Training is conducted with a batch size of 64 for 1000 steps, with gradient accumulation over 1 step, and a maximum sequence length of 512 tokens. The Adam optimizer is employed with a learning rate of 2×10^{-4} and a linear warm-up over the first 300 steps.

A.3 Training Time

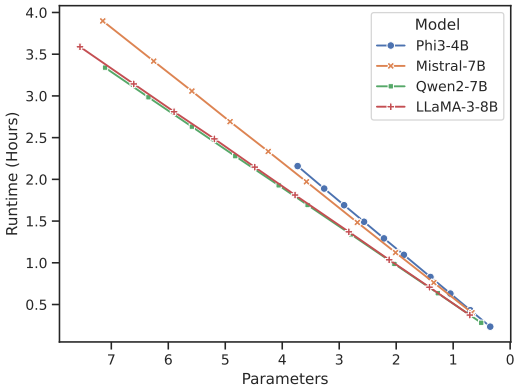

Figure 6: The total training time taken for all models at different pruning percentages.

Table 2 shows the time taken to train the two variants (`large` and `small`) provided by L^{3}Prune for each model. Figure 6 shows the training time for the models pruned at different pruning percentages, with respect to total parameter count. As we expect, the time taken to train a pruned model is linear to the pruning percentage, and corresponds roughly to the total parameter count. All models were trained

[2] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

	Large	Small
LLaMA-3-8B	2h 48m	35m
Mistral-7B	2h 41m	56m
Qwen2-7B	3h 1m	1h 14m
Phi3-4B	1h 40m	33m

Table 2: Training time for the variants produced by L^3Prune.

on a single NVIDIA A100 GPU. Including evaluation, we estimate that all experiments took a total of 200 GPU hours.

B Massive Text Embeddings Benchmark (MTEB)

B.1 MTEB subset details

MTEB encompasses a diverse array of embedding tasks varying in size, making a full evaluation quite time-consuming—it takes over 160 hours for a full-sized 7B model, such as Qwen2-7B, on an A100 GPU. To expedite our analysis, we use a representative subset of 15 tasks from MTEB, selected and used by BehnamGhader et al. (2024), detailed in Table 3. This subset includes tasks from each category in proportions closely matching those of the full MTEB.

Category	Dataset
Retrieval (3)	SciFact ArguAna NFCorpus
Reranking (2)	StackOverflowDupQuestions SciDocsRR
Clustering (3)	BiorxivClusteringS2S MedrxivClusteringS2S TwentyNewsgroupsClustering
Pair Classification (1)	SprintDuplicateQuestions
Classification (3)	Banking77Classification EmotionClassification MassiveIntentClassification
STS (3)	STS17 SICK-R STSBenchmark
SummEval (0)	-
Overall	15 datasets

Table 3: MTEB-15, the subset of MTEB tasks used for our work.

B.2 MTEB instructions

For evaluation on MTEB-15, we use the instructions from Wang et al. (2024), also used by BehnamGhader et al. (2024). The list of instructions for each task is listed in Table 4.

C Licenses

All four models we used are available for research purposes—LLaMA-3-8B is under its own permissive license, Mistral-7B and Qwen2-7B are under Apache License 2.0, and Phi3-4B is under MIT License. MTEB and the tasks it includes are provided under the Apache License 2.0. We overview the licenses of all datasets used in training below:

- ELI5: Provided under no specified license, available for research purposes.
- HotpotQA: Provided under CC BY-SA 4.0.
- FEVER: Provided under CC BY-SA 3.0.
- MIRACL: Provided under Apache License 2.0.
- MS-MARCO: Provided under no specific license, available for non-commercial research purposes.
- Natural Questions (NQ): Provided under CC BY 4.0.
- Stanford Natural Language Inference (SNLI): Provided under CC BY-SA 4.0.
- Multi Natural Language Inference (MNLI): Provided under a combination of permissive licenses, elaborated by Williams et al. (2018).
- SQuAD: Provided under CC BY-NC 4.0.
- TriviaQA: Provided under Apache License 2.0.
- Quora Duplicate Questions: Provided under no specified license, available for non-commercial purposes.
- Mr. TyDi: Provided under Apache License 2.0
- DuReader: Provided under Apache License 2.0
- T2Ranking: Provided under Apache License 2.0

Task Name	Instruction
Banking77Classification	Given a online banking query, find the corresponding intents
EmotionClassification	Classify the emotion expressed in the given Twitter message into one of the six emotions: anger, fear, joy, love, sadness, and surprise
MassiveIntentClassification	Given a user utterance as query, find the user intents
BiorxivClusteringS2S	Identify the main category of Biorxiv papers based on the titles
MedrxivClusteringS2S	Identify the main category of Medrxiv papers based on the titles
TwentyNewsgroupsClustering	Identify the topic or theme of the given news articles
SprintDuplicateQuestions	Retrieve duplicate questions from Sprint forum
SciDocsRR	Given a title of a scientific paper, retrieve the titles of other relevant papers
StackOverflowDupQuestions	Retrieve duplicate questions from StackOverflow forum
ArguAna	Given a claim, find documents that refute the claim
NFCorpus	Given a question, retrieve relevant documents that best answer the question
SciFact	Given a scientific claim, retrieve documents that support or refute the claim
STS*	Retrieve semantically similar text.

Table 4: Instructions used for evaluation on the MTEB benchmark. "STS*" refers to all the STS tasks.

Dataset	Instruction(s)
SNLI & MNLI	Given a premise, retrieve a hypothesis that is entailed by the premise Retrieve semantically similar text
DuReader	Given a Chinese search query, retrieve web passages that answer the question
ELI5	Provided a user question, retrieve the highest voted answers on Reddit ELI5 forum
FEVER	Given a claim, retrieve documents that support or refute the claim
HotpotQA	Given a multi-hop question, retrieve documents that can help answer the question
MIRACL	Given a question, retrieve Wikipedia passages that answer the question
MrTyDi	Given a question, retrieve Wikipedia passages that answer the question
MSMARCO Passage	Given a web search query, retrieve relevant passages that answer the query
MSMARCO Document	Given a web search query, retrieve relevant documents that answer the query
NQ	Given a question, retrieve Wikipedia passages that answer the question
QuoraDuplicates	Given a question, retrieve questions that are semantically equivalent to the given question Find questions that have the same meaning as the input question
SQuAD	Retrieve Wikipedia passages that answer the question
T2Ranking	Given a Chinese search query, retrieve web passages that answer the question
TriviaQA	Retrieve Wikipedia passages that answer the question

Table 5: Instructions used for each of the E5 datasets.