MediSwift: Efficient Sparse Pre-trained Biomedical Language Models

Vithursan Thangarasa[†] **Mahmoud Salem** **Shreyas Saxena**[*]
Kevin Leong **Joel Hestness** **Sean Lie**
Cerebras Systems
vithu@cerebras.net

Abstract

Large language models (LLMs) are typically trained on general source data for various domains, but a recent surge in domain-specific LLMs has shown their potential to outperform general-purpose models in domain-specific tasks (e.g., biomedicine). Although domain-specific pre-training enhances efficiency and leads to smaller models, the computational costs of training these LLMs remain high, posing budgeting challenges. We introduce MediSwift, a suite of biomedical LMs that leverage sparse pre-training on domain-specific biomedical text data. By inducing up to 75% weight sparsity during the pre-training phase, MediSwift achieves a 2-2.5x reduction in training FLOPs. Notably, all sparse pre-training was performed on the Cerebras CS-2 system, which is specifically designed to realize the acceleration benefits from unstructured weight sparsity, thereby significantly enhancing the efficiency of the MediSwift models. Through subsequent dense fine-tuning and strategic soft prompting, MediSwift models outperform existing LLMs up to 7B parameters on biomedical tasks, setting new benchmarks w.r.t efficiency-accuracy on tasks such as PubMedQA. Our results show that sparse pre-training, along with dense fine-tuning and soft prompting, offers an effective method for creating high-performing, computationally efficient models in specialized domains.

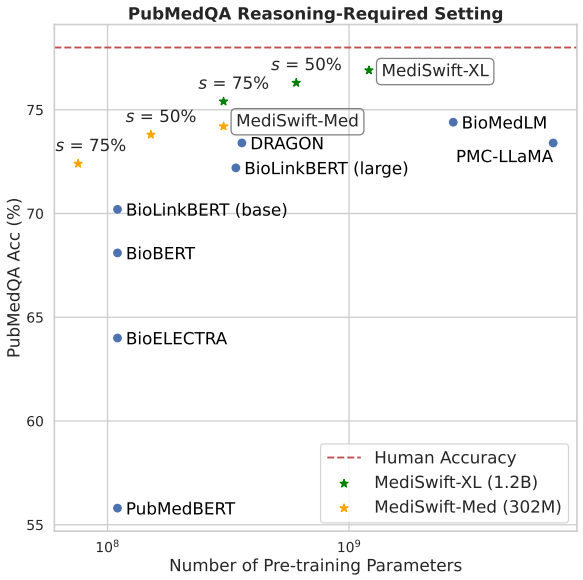

Figure 1: **Comparison of Model Size vs. PubMedQA Accuracy in the Reasoning-Required Setting:** Our dense and sparse MediSwift models noticeably outperform other fine-tuned language models $\leq$ 7B parameters, improving the efficiency-accuracy pareto frontier. In particular, MediSwift-XL (1.2B) achieves new state-of-the-art 76.8% accuracy at this size (i.e., being 5.8x smaller than PMC-LlaMA). In addition, sparse pre-trained MediSwift-XL models at $s \in \{50\%, 75\%\}$ outperform other models at similar or larger size. Additional details are provided in Table 2 and 3.

1 Introduction

The landscape of large language models (LLMs) has been predominantly shaped by efforts aimed at creating generalized models (Achiam et al., 2023; Touvron et al., 2023a; Zhang et al., 2022; Shoeybi et al., 2020), trained on diverse datasets that encompass a wide array of topics and domains, such as Pile (Gao et al., 2020), MassiveText (Hoffmann et al., 2022) and RedPajama (Computer, 2023). While these comprehensive datasets have included data in specialized domains (e.g., programming code (Chen et al., 2021) and PubMed Central (Gao et al., 2020)), the overarching goal has been to forge LLMs capable of broad applicability. However, recent efforts in training models on domain-specific data are emerging, with these smaller, specialized models surpassing general-purpose ones in domain focused tasks, especially in science (Taylor et al., 2022) and medicine (Luo et al., 2022; Bolton et al., 2021). This has sparked a renewed interest in the

[*]Work done while at Cerebras.
[†]Corresponding author.

development of LLMs tailored to specific domains, suggesting a promising avenue for enhancing compute efficiency and model performance w.r.t evaluation metrics on downstream tasks.

Furthermore, the shift towards specialized LLMs in the field of medicine is particularly gaining recognition for its capacity to significantly improve the accuracy and effectiveness of these models. This is achieved by closely aligning them with the specific needs and complexities of this specialized area. For example, in biomedicine, models trained on specialized literature (e.g., MEDITRON (Chen et al., 2023), BioGPT (Luo et al., 2022), BioMedLM (Bolton et al., 2021)) exhibit significant improvements over general ones. This approach involves either continued pre-training on domain-relevant texts or pre-training models from scratch with domain-specific data (e.g., PubMed Central[1] and PubMed open-access research papers), emphasizing the accuracy benefits and ability to yield models that are more compute efficient.

After pre-training models on domain-specific data, further accuracy enhancements are achieved through prompt-based fine-tuning (Nori et al., 2023b; Reynolds and McDonell, 2021; Peng et al., 2023; Liu et al., 2022b) and full fine-tuning of the weights (Ziegler et al., 2019; Cohen et al., 2022; Hu et al., 2021), offering a balance between adaptability, efficiency, and task-specific optimization. This dual approach merges the model's pre-trained knowledge with the specific needs of downstream tasks, thereby facilitating a more accurate and effective application of the model's capabilities across a wide array of downstream applications; maximizing the potential of domain-specific LLMs.

Despite these advancements and the clear benefits of domain-specific LLM pre-training, the computational demands and associated costs remain a significant challenge, highlighting the need for innovative solutions to make these powerful tools more accessible and efficient for a wide-range of applications. In response, a variety of techniques have been proposed aimed at mitigating the computational burden associated with training LLMs. These include sparse attention (Dao et al., 2022b; Jaszczur et al., 2021), quantized optimization (Tang et al., 2021), low-rank factorization (Lialin et al., 2023) and sequence-level curriculum learning (Li et al., 2022). Among these approaches, weight sparsity emerges as a particularly promising method, distinct from the others in its approach by setting a subset of model parameters to zero, thus reducing the FLOPs needed during training.

However, the adoption of sparse training is limited by (1) the challenge of finding optimal sparsity patterns which retain the accuracy of the original dense model (Frankle and Carbin, 2018; Ma et al., 2022) and (2) its difficulties in accelerating on hardware optimized for dense computations (e.g., GPUs and TPUs) (Hooker, 2020). Additionally, sparse pre-training of LLMs typically leads to an accuracy loss from optimization challenges in sparse networks (Evci et al., 2019), but previous studies have shown that transitioning from sparse-to-dense training can effectively address these problems (Thangarasa et al., 2023a; Dao et al., 2022a). Our work aims to reduce training FLOPs by employing unstructured weight sparsity in domain-specific LLMs and recovering lost representational capacity by transitioning to dense weight matrices for fine-tuning on downstream tasks.

Despite weight sparsity not being widely used in real-world applications, advancements in specialized software kernels (Gale et al., 2019; Elsen et al., 2019; Ashby et al., 2019; Wang, 2021; Tang et al., 2023) have facilitated inference acceleration with unstructured sparsity. Recent developments have shown that the benefits of unstructured weight sparsity can be fully harnessed on specialized hardware, such as the Cerebras CS-2 (Cerebras, 2023; Lie, 2022b), for LLM training. As sparse training techniques and hardware continue to co-evolve, we anticipate that the reductions in FLOPs will lead to realized sparse acceleration. The latest innovations in software and hardware (NeuralMagic, 2021; Gupta, 2024) are geared towards enabling the widespread adoption of unstructured weight sparsity, offering the potential to achieve higher compression ratios and practical speedups in terms of wall-clock time.

Building on this momentum, we introduce MediSwift, a suite of biomedical language models (LLMs) available in three sizes: Med (302M), Large (510M), and XL (1.2B). These models are based on GPT-3 and pre-trained sparsely from scratch on biomedical texts, aimed at reducing the computational costs required for training. We explore the impact of applying 50% and 75% weight sparsity during pre-training, which results in a 2-2.5x reduction in the overall training FLOPs needed. Figure 1 summarizes the performance, where our dense and sparse MediSwift models no-

[1]https://pubmed.ncbi.nlm.nih.gov/

ticeably outperform other language models up to 7B parameters. We demonstrate MediSwift's capabilities through fine-tuning on established benchmarks for biomedical natural language processing (NLP) tasks (e.g., PubMedQA (Jin et al., 2019) for question answering and HoC (Baker et al., 2016) for document classification) showing significant improvements in the balance between efficiency and accuracy. Although previous research suggests that sparse pre-training may compromise model accuracy on downstream tasks, our approach incorporates dense fine-tuning with strategic soft prompting to effectively regain performance on specialized tasks. Specifically, MediSwift-XL (1.2B) sets a new state-of-the-art by reaching 76.8% accuracy, despite being 5.8x smaller than PMC-LlaMA. Moreover, MediSwift-XL models, pre-trained with 50% and 75% sparsity, surpass the performance of models of similar or greater sizes (e.g., MediSwift-XL at 75% sparsity outperforms BioMedLM while being almost 9x smaller). Our work not only highlights the potential for sparse pre-training to make LM training more economically viable but also sets a new benchmark for efficiency in domain-specific applications of LLMs. The key contributions are:

1. We introduce MediSwift, a family of biomedical language models in three sizes (Med, Large, and XL), and extend this by introducing both dense and sparse variants pre-trained with 50% and 75% weight sparsity. This diversification balances computational efficiency with model effectiveness in biomedical applications, offering options for different computational resource needs.

2. To our knowledge, this is the first study to highlight the benefits of sparse pre-training on biomedical texts, achieving significant computational savings. We show inducing 75% weight sparsity into MediSwift models, results in a 2.5x reduction in training FLOPs, while improving efficiency-accuracy trade-offs in tasks like PubMedQA.

3. We demonstrate that despite the potential for sparse pre-training to reduce model accuracy, dense fine-tuning combined with soft prompting can effectively regain performance on task-specific fine-tuning. Specifically, 50% sparse MediSwift-XL achieves a new state-of-the-art with 76.3% accuracy on PubMedQA, surpassing existing models up to 7B parameters.

2 Methodology

In this section, we formalize our two-phase training framework for MediSwift models to reduce computational costs and yet retain model accuracy. Initially, we pre-train these models on biomedical data, applying unstructured weight sparsity to reduce the computational training FLOPs. Following this, we enhance the model through dense fine-tuning, reactivating weights to improve adaptability for specific tasks, and incorporate soft prompting to refine responses for task requirements. This efficient approach, combining sparse pre-training, dense fine-tuning, and soft prompting, significantly boosts both model efficiency and performance, as our results demonstrate in Section 4.

2.1 Autoregressive Language Modeling

Dense Pre-training Autoregressive LMs predict a series of tokens by making each token's prediction dependent on the ones before it, similar to a Markov chain process. This method follows core principles of language modeling, aiming to understand the pattern of token sequences unsupervisedly from a corpus of text data. Consider an unsupervised corpus $\mathcal{U}$ consisting of tokens $u_1, u_2, \ldots, u_{|\mathcal{U}|}$, with $|\mathcal{U}|$ denoting the corpus's total token count. Our objective is to enhance the model's ability to predict sequences by maximizing the likelihood of the observed sequences, formulated as the sum of the log probabilities of each token given its preceding context within a window of size k. The mathematical representation of this objective is as follows:

$$\mathcal{L}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{U}|} \log(p(u_i|u_{i-k:i-1}, \theta)),$$

where θ denotes the neural network's parameters, encapsulating the *dense* configuration of the network's architecture. The context window, k, determines the number of preceding tokens used for current token prediction. The neural network, parameterized by $\theta \in \mathbb{R}^N$, where N is the total parameter count, aims to optimize these parameters across all layers L, with each layer l having its own set of parameters θ_l.

Sparse Pre-training Building upon our framework for dense pre-training, we introduce weight sparsity into the model, specifically to improve the computational efficiency. We achieve this by

methodically reducing the number of active connections within each layer l of the model by a predefined sparsity level $s_l \in (0, 1)$, effectively rendering a portion of the network's parameters inactive. This process yields a network with $(1 - s_l)N_l$ active parameters per layer, where N_l denotes the original number of parameters in layer l. The overall sparsity is quantified by S, which represents the ratio of inactive parameters to the total parameter count of the initially dense model, calculated as $S = \frac{\sum_l^L s_l N_l}{N}$. To apply sparsity effectively, we employ a binary mask $m \in \{0, 1\}^{|\theta|}$ to the model's initial parameters θ^0, resulting in a sparse parameter set $m \odot \theta^0$. This mask effectively segregates the parameters into active (1) and inactive (0) states, thereby establishing a sparsity-induced version of the language model that aims to minimize a slightly modified objective:

$$\mathcal{L}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{U}|} \log(p(u_i|u_{i-k:i-1}, m \odot \theta)). \quad (1)$$

We leverage the GPT-3 (Brown et al., 2020a) architecture for the MediSwift biomedical language model, training it with the AdamW (Loshchilov and Hutter, 2017) optimizer on a carefully curated biomedical dataset following the objective shown in Eq. 1 for j iterations to get parameters θ^j. This pre-trained model is then fine-tuned for specific supervised tasks in the biomedical domain.

2.2 Dense Fine-tuning and Soft Prompting

In this section, we detail the adaptation of our pre-trained MediSwift model for tasks like biomedical question answering (QA) and document classification using dense fine-tuning and soft prompting. We align tasks with varying output formats to our pre-training format by converting task labels into natural language sequences (Hu et al., 2021; Li and Liang, 2021). This method avoids structured formats and special tokens, ensuring semantic coherence and making full use of the natural language corpus MediSwift was trained on.

Following Luo et al. (2022), each downstream fine-tuning task is represented by a training dataset consisting of `source-target` pairs defined as: $\mathcal{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{|x|}, y_{|y|})\}$, where both x and y are sequences of tokens. For example, in question answering (e.g., PubMedQA), x corresponds to the question and reference context description, and y the categorical answer to the question; in biomedical document classification (e.g., Hallmarks of Cancers corpus), x is the text passage and y corresponds to the hallmarks of cancer.

2.2.1 Dense Fine-tuning

We begin fine-tuning with parameters θ^j set at their pre-trained values, adjusting them by a task-specific increment $\Delta\theta$ with the same dimensionality, $|\Delta\theta| = |\theta|$. Unlike prior efforts that aimed at parameter efficiency for easier model deployment (Hu et al., 2021; Dettmers et al., 2023), we prioritize reducing pre-training computational costs via unstructured weight sparsity and enhance network representation by adopting dense fine-tuning (Thangarasa et al., 2023a). This approach overcomes sparse optimization challenges by reactivating previously inactive weights during the dense fine-tuning phase, thus enhancing the model's capacity. By removing the sparsity mask m, we allow $\sum_l^L s_l \cdot N_l$ weights to be reactivated and initialize them to zero—a method proven more effective than other initialization strategies (e.g., scaled uniform or normal distribution initializations (Evci et al., 2020)). The network is then densely updated to optimize the loss function:

$$\mathcal{L}(\mathcal{Z}) = \sum_{(x,y)\in\mathcal{Z}} \log \prod_{t=1}^{|y|} p(y_t|(x_1, \ldots, x_{t-1}), \theta^j + \Delta\theta) \quad (2)$$

2.2.2 Soft Prompting

Previous research on language models in the biomedical field has mainly focused on fine-tuning for domain-specific tasks (Bolton et al., 2021). Recently, there has been a move towards improving biomedical NLP task performance through prompt engineering (Luo et al., 2022; Nori et al., 2023b; Yagnik et al., 2024). Drawing inspiration from these advancements, our approach integrates prompt-based techniques into the fine-tuning phase of MediSwift. More precisely, we adopt the soft prompting methodology as described by Luo et al. (2022), aiming to refine our model's capability in understanding and processing biomedical text. Similar to prefix-tuning (Li and Liang, 2021), to integrate soft prompts, we insert virtual tokens between the source and target sequences, thus modifying the loss function $\mathcal{L}(\mathcal{Z})$ in Eq. 2. This adjustment accounts for the `[source; prompt; target]` sequence structure, impacting the model's learning and inference. Let $\mathcal{P}$ denote the prompt, consisting

of a sequence of virtual tokens $\in \{v_1, v_2, \ldots, v_n\}$, where n is the number of virtual tokens, and these tokens are represented by continuous embeddings. The modified Eq. 2, reflecting the inclusion of the prompt and its positioning, can be formalized as:

$$\mathcal{L}(\mathcal{Z}) = \sum_{(x,y)\in\mathcal{Z}} \log \prod_{t=1}^{|\boldsymbol{y}|} p(y_t|([x;\mathcal{P}];y_{<t}),\theta^j + \Delta\theta) \quad (3)$$

Through this multi-faceted approach, our pre-training and fine-tuneing method for MediSwift not only addresses the computational efficiency challenges in LM training, but also leverages the capabilities of in-domain pre-trained LMs to improve performance on biomedical NLP tasks.

3 MediSwift Biomedical Pre-training

This section describes the MediSwift pre-training process, including data sources, collection, and preprocessing for biomedical data. We explain the dataset's origins, statistical analysis, and preparation for efficient training. We also compare MediSwift models' performance with both dense and sparse pre-training, emphasizing training convergence differences and FLOPs savings.

3.1 PubMed Papers and Abstracts

MediSwift is an in-domain biomedical language model, drawing its strength from an exclusive pre-training regimen focused solely on biomedical textual data. Its foundation lies in the vast repository of available open-access medical research papers and abstracts found in PubMed Central (PMC) (National Library of Medicine. 2003–2023), similar to the approaches used in prior models (e.g., Meditron (Chen et al., 2023), BioGPT (Luo et al., 2022), BioMedLM (Bolton et al., 2021)).

PMC consists of 4.91M full-text papers, and PubMed and PMC Abstracts comprise of 16.1 million papers (see Table 1). Moreover, we gathered the most recent PubMed entries, updated prior to 2023, directly from the official website[1], utilizing the official scripts for PubMed Abstracts[2] and PubMed Central[3]. Similar to Luo et al. (2022) and Chen et al. (2023), we filter out empty items containing solely titles without accompanying abstracts.

[2] https://github.com/thoppe/The-Pile-PubMed?tab=readme-ov-file

[3] https://github.com/EleutherAI/pile-pubmedcentral

Table 1: Statistics on the mixture of pre-training data for MediSwift, including the sizes of the training and validation sets. Total sample count is provided for each set, along with the percentage of validation set allocation relative to the training set.

Dataset	Number of Samples	
	Train	Validation
PubMed Abstracts[1]	15.7M	487K (3%)
PubMed Papers[1]	4.9M	142K (3%)
Total	20.6M	629K

Furthermore, prior works have show the significance of in-domain vocabulary for improving performance of specialized language models (Gu et al., 2021; Wu et al., 2023b; Mielke et al., 2021), a critical step that is often overlooked. Therefore, inspired by Luo et al. (2022) and (Bolton et al., 2021), instead of using the standard GPT-3 vocabulary, we learned the vocabulary directly from the biomedical corpus. Employing Moses (Koehn et al., 2007) tokenization followed by byte pair encoding (BPE), we segment the corpus into word pieces and learn the vocabulary; resulting in a size of 42,384. By exclusively pre-training with biomedical texts and using a specialized vocabulary, MediSwift improves the efficiency-accuracy frontier, as empirically shown through fine-tuning on biomedical tasks in Section 4.

3.2 MediSwift Pre-training

Pre-training Experimental Details We pre-train and benchmarked MediSwift in-domain biomedical language models at 3 sizes: 302M, 510M and 1.21B. All MediSwift models are pre-trained from scratch using the Cerebras CS-2[4], taking advantage of its ability to accelerate training with unstructured sparsity. At the time of the study, the specialized kernels of Cerebras CS-2 were designed to facilitate training with static unstructured sparsity (refer to Appendix B for additional details regarding the hardware accelerator).

In the pre-training phase of the MediSwift models, a couple sparsity levels are explored, including 50%, and 75%, aside from their respective dense counterparts. The pre-training of MediSwift models are conducted on a single CS-2 for a total of 200,000 steps, with a batch size of 512 and a maximum sequence length of 1024 tokens, resulting

[4] https://docs.cerebras.net/en/2.1.1/wsc/how_to_guides/sparsity.html

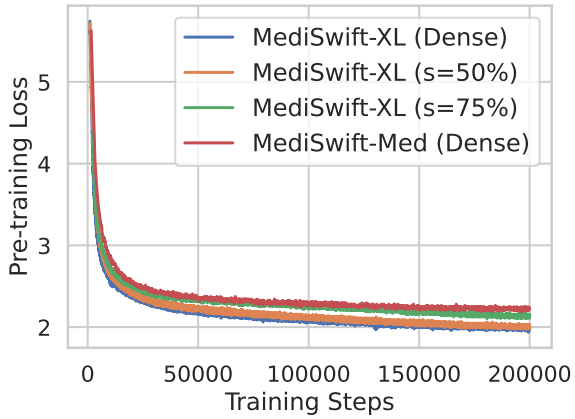

Figure 2: **Comparison of pre-training loss curves for MediSwift models**: MediSwift-XL's training loss reveals that at 50% sparsity, the model's performance closely mirrors that of its dense variant, with negligible effects on training loss. At 75% sparsity, although the gap in training loss widens, the sparse MediSwift-XL still outperforms the dense MediSwift-Med, showcasing efficient learning even at higher sparsity levels.

in approximately 104.86B tokens processed in total (see Appendix A.1 for additional pre-training experimental setup details).

3.3 Sparse Pre-trained MediSwift

While there are several advanced sparse training techniques (Evci et al., 2020; Mocanu et al., 2018; Liu et al., 2021), for simplicity, in this work, we adopt static sparsity, namely random pruning, for the sparse pre-training of MediSwift models. This approach mandates a uniform distribution of sparsity levels throughout each layer, irrespective of the layer's parameter count or its FLOPs. Specifically, the scope of our sparsification process is confined to all dense linear layers within the network, including both matrices within the multilayer perceptron (MLP) module—namely, the intermediate layer and the MLP output projection, as well as the four weight matrices integral to the self-attention (Vaswani et al., 2017) mechanism: query, key, value, and attention output projection. Notably, we ensure that the embeddings, Layer Normalization (Ba et al., 2016) components and biases are kept dense.

3.4 Analysis on Pre-trained Models

In Figure 2, we illustrate the training loss curves for the MediSwift-XL model for both the dense and sparse configurations at $s \in \{50\%, 75\%\}$. The 50% sparse model's training loss closely follows that of the dense MediSwift-XL, showcasing minimal deviation throughout the training process. However, a noticeable divergence is observed at 75% sparsity, where the final training loss slightly lags behind that of the dense counterpart. Interestingly, when comparing the 75% sparse MediSwift-XL model with the dense MediSwift-Med model, the former, despite sharing the same number of non-embedding parameters, it achieves a lower training loss (refer to Table 2). This observation aligns with previous findings that larger, albeit sparser, models can surpass their smaller, densely parameterized equivalents in terms of performance (Thangarasa et al., 2023b; Liu et al., 2022a; Ramanujan et al., 2020; Golubeva et al., 2021). This superiority is further corroborated by the improved performance of the 75% sparse model on biomedical NLP tasks, when compared to the MediSwift-Med dense model (see Section 4), highlightling the benefits of training larger sparse models in comparison to smaller dense ones.

Table 2: **Final Pre-training Losses and Computational Efficiency of MediSwift Models.** We summarize the results for the MediSwift-Med and MediSwift-XL models, trained using the biomedical pre-training corpus detailed in Section 3.1. We report the final pre-training losses for dense and sparse variants at 50% and 75% sparsity levels. The total FLOPs and FLOP savings relative to the dense baseline are indicated in parentheses, highlighting the models' computational efficiency.

MediSwift	Size	Pre-train Loss	Train FLOPs (x 10^{20})
$\text{Med}_{\text{dense}}$	302M	2.234	2.677 (1.00x)
$\text{Med}_{s=50\%}$	151M	2.265	1.727 (0.64x)
$\text{Med}_{s=75\%}$	0.76M	2.375	1.252 (0.46x)
XL_{dense}	1.21B	1.979	9.148 (1.00x)
$\text{XL}_{s=50\%}$	605M	2.012	5.348 (0.58x)
$\text{XL}_{s=75\%}$	302M	2.141	3.448 (0.38x)

In Table 2, we provide an analysis of the computational efficiency achieved through sparse pre-training in our proposed MediSwift architectures, namely MediSwift-Med and MediSwift-XL. Here, we quantify the total FLOPs required for both forward and backward propagations during the pre-training phase of these models.

For the MediSwift-XL model, attention and vocabulary embeddings represent 13.3% and 6.8% of total FLOPs, respectively, hence highlighting the computational savings at the 1.2B parameter scale. Sparse pre-training with 75% sparsity reduces FLOPs by slightly over 2.5x compared to

its dense counterpart. The smaller MediSwift-Med model has a higher FLOP percentage for attention and embeddings, hence achieving a 2x reduction at the same sparsity level. This indicates that FLOP savings increase with model size, demonstrating that larger models (Hoffmann et al., 2022; Kaplan et al., 2020; Hestness et al., 2017) can potentially benefit more from sparse pre-training. In addition, we emphasize that the total FLOPs required for fine-tuning these models account for a minor fraction of the overall pre-training FLOPs, reinforcing the efficiency of our approach in scaling to larger model sizes while conserving computational resources.

4 Fine-tuning on Biomedical tasks

This section evaluates MediSwift's performance on PubMedQA and HoC benchmarks using dense fine-tuning on each variant's specific training set (e.g., fine-tuning and testing with the PubMedQA dataset). During the fine-tuning phase, the total training FLOPs constitute only a minor fraction of the FLOPs expended during pre-training, despite the fine-tuning being conducted densely. As a result, the FLOPs consumed during sparse pre-training is proportional to the combined FLOPs of sparse pre-training and dense fine-tuning.

Following Luo et al. (2022), we incorporate soft prompting into our fine-tuning framework by formatting sequences as `[source; prompt; target]`. This format helps our models better utilize contextual information, demonstrating MediSwift's effectiveness and adaptability in medical text analysis. Further details on hyperparameters and dataset specifics are provided in Appendix A.2 and A.3. We note that all fine-tuning results were averaged across 3 random seeds.

4.1 Question Answering with PubmedQA

We assess MediSwift's performance on the PubMedQA (Jin et al., 2019) dataset, which is derived from PubMed abstracts and includes three subsets: PQA-A, PQA-U, and PQA-L. We adhere to the original train/val/test splits, focusing on the PQA-L test set for the final evaluation. Our approach utilizes multi-stage fine-tuning and soft prompting with continuous embeddings of length $n = 9$, which was shown to perform the best in terms of accuracy (Luo et al., 2022). Following established preprocessing methods, we format the data into `[source, target]` sequences, each consisting of a question, reference context, long answer, and a categorical label, `[yes/no/maybe]`, for the answer. The performance is measured by classification accuracy, particularly under the challenging *reasoning-required setting* (Jin et al., 2019), where the model predicts based on the question and context without the long answer.

In Table 3, we demonstrate that across all sizes, MediSwift improves the pareto frontier in PubMedQA accuracy, notably with the dense MediSwift-XL model setting a new benchmark while being significantly smaller, at 5.8x less size

Table 3: MediSwift's performance on the PubMedQA reasoning-required task in both dense and sparse settings, $s \in \{50\%, 75\%\}$. This table compares MediSwift against other language models from the PubMedQA leaderboard, demonstrating its efficiency-accuracy improvement. Results are shown for models $\leq$ 7B parameters, with the "size" column indicating pre-training parameters and the final column reporting test accuracy on the PQA-L test set.

Model	Size	Acc.
PubMedBERT (Gu et al., 2021)	110M	55.8
BioELECTRa (Kanakarajan et al., 2021)	110M	64.2
BioLinkBERT$_{\text{base}}$ (Yasunaga et al., 2022b)	110M	70.2
BioLinkBERT$_{\text{large}}$ (Yasunaga et al., 2022b)	340M	72.2
BioGPT$_{\text{med}}$ (Luo et al., 2022)	345M	73.6†
DRAGON (Yasunaga et al., 2022a)	360M	73.4
MediSwift-Med$_{\text{dense}}$	302M	74.2
MediSwift-Med$_{(s=50\%)}$	151M	73.8
MediSwift-Med$_{(s=75\%)}$	0.76M	72.4
BioGPT$_{\text{large}}$ (Luo et al., 2022)	1.54B	75.5†
BioMedLM (Bolton et al., 2021)	2.70B	74.4
PMC-Llama (Wu et al., 2023a)	7.00B	73.4
GPT-3.5 (0-shot) (Nori et al., 2023a)	-	71.6
GPT-4 (0-shot) (Nori et al., 2023a)	-	75.2
MediSwift-Large$_{\text{dense}}$	510M	75.1
MediSwift-Large$_{(s=50\%)}$	255M	74.2
MediSwift-Large$_{(s=75\%)}$	128M	73.4
MediSwift-XL$_{\text{dense}}$	**1.21B**	**76.8**
MediSwift-XL$_{(s=50\%)}$	605M	76.3
MediSwift-XL$_{(s=75\%)}$	302M	75.4

† We followed the fine-tuning steps used in the official BioGPT code[5] to reproduce the results on BioGPT$_{\text{med}}$ and BioGPT$_{\text{large}}$, which reported accuracies of 78.2% and 81.0%, respectively (Luo et al., 2022). However, the methodologies for fine-tuning on PubMedQA (Luo et al., 2022), as well as the fine-tuning scripts, lack clear descriptions and details, making it difficult to reproduce these results, especially under a *reasoning-required setting*. Hence, we made efforts to replicate their findings as closely as possible, despite uncertainties about the original experimental setup.

[5] https://github.com/microsoft/BioGPT/tree/main/examples/QA-PubMedQA

than PMC-LLaMA. This trend continues with the 50% and 75% sparse variants of MediSwift, which surpass other language models of comparable or larger sizes. Specifically, the 75% sparse MediSwift-XL exceeds BioMedLM's performance by 1.0% while being approximately 8.9x smaller. Furthermore, within the MediSwift family, the larger yet sparse 75% MediSwift-XL demonstrates superior performance over the smaller dense MediSwift-Med by 1.2%, despite both models sharing the same pre-training parameters.

4.2 Document Classification on HoC

We examine the Hallmarks of Cancers (HoC) corpus (Baker et al., 2016), comprising 1580 PubMed abstracts annotated for ten cancer hallmarks. We tackle a document classification task, assigning documents to predefined single or multi-label categories, and using MediSwift to generate label words. We follow the established train/dev/test splits of 1108/157/315 (Gu et al., 2021). Similar to Luo et al. (2022), we employ a continuous embedding of length $n = 1$ as the prompt, and we incorporate labels into the target sequence.

The performance is evaluated using the micro-F1 score, allowing direct comparison with prior models and demonstrating our method's effectiveness. In Table 4, the dense MediSwift-XL model outperformed all similarly sized models in micro-F1 score, with its 50% and 75% sparse demonstrating very competitive results, emphasizing sparse pre-training's balance of computational efficiency and accuracy. This further showcases the potential of sparsity in optimizing language model performance for biomedical applications.

Table 4: MediSwift's performance on the Hallmarks of Cancers (HoC) document classification task in both dense and sparse settings, $s \in \{50\%, 75\%\}$. This table compares MediSwift against other language models, demonstrating its efficiency-accuracy improvement. The "size" column indicating pre-training parameters and the final column reporting micro-F1 score on the test set.

Model	Size	F1
BioBERT (Lee et al., 2019)	110M	81.54
PubMedBERT (Gu et al., 2021)	110M	82.32
BioLinkBERT$_{\text{base}}$ (Yasunaga et al., 2022b)	110M	84.35
BioLinkBERT$_{\text{large}}$ (Yasunaga et al., 2022b)	340M	84.57
GPT-2$_{\text{med}}$ (Luo et al., 2022)	345M	81.54
BioGPT$_{\text{med}}$ (Luo et al., 2022)	345M	85.12
BioGPT$_{\text{large}}$ (Luo et al., 2022)	1.54B	84.40
MediSwift-Med$_{\text{dense}}$	302M	85.15
MediSwift-Med$_{\text{s=50\%}}$	151M	84.48
MediSwift-Med$_{\text{s=75\%}}$	0.76M	83.95
MediSwift-Large$_{\text{dense}}$	510M	85.22
MediSwift-Large$_{\text{s=50\%}}$	255M	84.63
MediSwift-Large$_{\text{s=75\%}}$	128M	84.12
MediSwift-XL$_{\text{dense}}$	**1.21B**	**85.46**
MediSwift-XL$_{\text{s=50\%}}$	605M	84.98
MediSwift-XL$_{\text{s=75\%}}$	302M	84.71

5 Related Work

Sparse Training for Language Models Sparse weight training for language models has emerged as a promising avenue to address the computational intensity of training large models. Recent studies have explored various sparse training methodologies (Thangarasa et al., 2023a; Dao et al., 2022a; Chen et al., 2022), aiming to maintain or enhance model performance while significantly reducing computational requirements. Techniques such as pruning (Chen et al., 2020), sparse activations (Mirzadeh et al., 2024), along with the development of specialized software (NeuralMagic, 2021; Gupta, 2024) and hardware (Lie, 2022b,a; Dietrich et al., 2021) have been pivotal. Our work builds on these foundations, focusing on optimizing sparse weight training strategies specifically for domain-specific language models, pushing the boundaries of efficiency.

Biomedical Language Models The evolution of language models for medical applications has progressed from adapting encoder-only architectures like BERT (Devlin et al., 2018), using biomedical corpora (Lee et al., 2019; Gu et al., 2021), to incorporating strategies like document links (Yasunaga et al., 2022b) and knowledge graphs (Yasunaga et al., 2022a). The shift towards autoregressive generative models, such as GPT (Brown et al., 2020b) and Llama (Touvron et al., 2023a), for pretraining on medical texts has led to significant advancements (Wu et al., 2023a; Luo et al., 2022; Bolton et al., 2021). Recent scaling efforts include GatorTronGPT with 20B parameters (Yang et al., 2022), as well as Clinical-Camel (Toma et al., 2023), MEDITRON (Chen et al., 2023) and Med-42 (Christophe et al., 2023), based on Llama-2-70B (Touvron et al., 2023b), focusing on mixed clinical and general English texts.

Our work diverges from works that scaled up

medical LMs by introducing weight sparsity into the pre-training of biomedical LMs. This reduces the computational costs typically associated with large-scale models, thereby improving the balance between efficiency and accuracy in the medical domain.

Prompting for Biomedical Language Models Recent research has shifted towards prompt engineering to enhance language models' performance on biomedical tasks, such as BioGPT's (Luo et al., 2022) use of soft prompt-tuning and Medprompt's (Nori et al., 2023b) innovative prompting techniques for generalist foundation models. Liévin et al. (2023) and Yagnik et al. (2024) analyzed the effectiveness of prompting in the medical domain and showed that it can improve metric scores. However, combining task-specific fine-tuning with prompting strategies, as seen in Med-PaLM 2 (Singhal et al., 2022, 2023), yields competitive results on challenging biomedical tasks. Our work extends this by integrating task-specific fine-tuning and soft prompting to address model accuracy loss during sparse pre-training, effectively achieving efficiency gains with minimal accuracy degradation on biomedical tasks.

6 Conclusion

In conclusion, MediSwift innovates in biomedical language models by combining sparse pre-training with dense fine-tuning and soft prompting, balancing computational efficiency with accuracy. Available in Med, Large, and XL sizes, with 50% and 75% sparsity, MediSwift addresses the cost of training models and sets new standards for biomedical tasks like PubMedQA. MediSwift-XL, in particular, showcases superior efficiency-accuracy trade-offs, outperforming models up to 7B parameters. This work exemplifies the potential of sparse pre-training as a cost-effective method for developing specialized, high-performance models, establishing MediSwift as a benchmark in biomedical NLP.

7 Limitations

Our work on MediSwift represents a significant leap forward in developing efficient domain-specific LLMs, particularly in biomedicine, by utilizing sparse pre-training to strike a fine balance between computational efficiency and accuracy. While we have initially focused on static sparse pre-training, the emerging field of dynamic sparse training (DST) holds great promise for further improvements (Evci et al., 2020; Mocanu et al., 2018; Liu et al., 2021).

DST offers an exciting avenue for optimizing sparsity patterns dynamically, potentially elevating model quality and training efficiency to new heights. Although the implementation of DST requires advanced software and hardware support for unstructured sparse computations—capabilities that were beyond our current scope—this innovative approach represents an interestingly opportunity for future research. As support for unstructured sparse training evolves with ML software-hardware co-design, we anticipate these advancements will enable us to harness DST, paving the way for even more high-quality and efficient domain-specific LLMs.

8 Ethical Considerations

While MediSwift represents a significant advancement in encoding medical knowledge from sources of high-quality evidence, it is important to note that it has not been fully adapted to deliver this knowledge in a manner that is appropriate, safe, or within the actionable constraints required by medical professionals. Therefore, we strongly recommend against deploying MediSwift directly in clinical or medical applications without thorough alignment with specific use cases.

Moreover, additional testing is crucial, including the conduct of randomized controlled trials in real-world practice settings, to ensure the model's recommendations are reliable and beneficial in practical healthcare environments. This cautionary approach emphasizes the importance of bridging the gap between technological capabilities and the nuanced requirements of medical practice to ensure patient safety and efficacy of care.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, and et al. 2023. Gpt-4 technical report. *arXiv*.

Mike Ashby, Christiaan Baaij, Peter Baldwin, Martijn Bastiaan, Oliver Bunting, Aiken Cairncross, Christo-

pher Chalmers, Liz Corrigan, Sam Davis, Nathan van Doorn, Jon Fowler, Graham Hazel, Basile Henry, David Page, Jonny Shipton, and Shaun. Steenkamp. 2019. Exploiting unstructured sparsity on next-generation datacenter hardware.

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

Simon Baker, Ilona Silins, Yufan Guo, Imran Ali, Johan Högberg, Ulla Stenius, and Anna Korhonen. 2016. Automatic semantic classification of scientific literature according to the hallmarks of cancer. *Bioinformatics*.

Elliot Bolton, David Hall, Michihiro Yasunaga, Tony Lee, Chris Manning, and Percy Liang. 2021. Biomedlm.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020a. Language models are few-shot learners. In *NeurIPS*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020b. Language models are few-shot learners. In *NeurIPS*.

Cerebras. 2023. Train a model with weight sparsity. *Cerebras Wafer-Scale cluster (R2.1.1) Documentation*.

Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. 2022. Pixelated butterfly: Simple and efficient sparse training for neural network models. In *ICLR*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *arXiv*.

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

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. 2023. Meditron-70b: Scaling medical pretraining for large language models. *arXiv*.

Clément Christophe, Avani Gupta, Nasir Hayat, Praveen Kanithi, Ahmed Al-Mahrooqi, Prateek Munjal, Marco Pimentel, Tathagata Raha, Ronnie Rajan, and Shadab Khan. 2023. Med42 - a clinical large language model.

Aaron Daniel Cohen, Adam Roberts, Alejandra Molina, Alena Butryna, Alicia Jin, Apoorv Kulshreshtha, Ben Hutchinson, Ben Zevenbergen, Blaise Hilary Aguera-Arcas, Chung ching Chang, Claire Cui, Cosmo Du, Daniel De Freitas Adiwardana, Dehao Chen, Dmitry (Dima) Lepikhin, Ed H. Chi, Erin Hoffman-John, et al. 2022. Lamda: Language models for dialog applications. In *arXiv*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. 2022a. Monarch: Expressive structured matrices for efficient and accurate training. In *ICML*.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. 2022b. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*.

Anastasia Dietrich, Frithjof Gressmann, Douglas Orr, Ivan Chelombiev, Daniel Justus, and Carlo Luschi. 2021. Towards structured dynamic sparse pre-training of bert.

Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. 2019. Fast sparse convnets. *arXiv*.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *ICML*.

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

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv*.

Anna Golubeva, Guy Gur-Ari, and Behnam Neyshabur. 2021. Are wider nets better given the same number of parameters? In *ICLR*.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Trans. Comput. Healthcare*.

Abhay Gupta. 2024. Sparsity made easy - introducing the cerebras pytorch sparsity library. *Cerebras Systems Blog*.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep learning scaling is predictable, empirically. *arXiv*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, Laurent Sifre, and et al. 2022. An empirical analysis of compute-optimal large language model training. In *NeurIPS*.

Sara Hooker. 2020. The Hardware Lottery. *arXiv*.

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

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. 2021. Sparse is enough in scaling transformers. In *NeurIPS*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *EMNLP-IJCNLP*.

Kamal raj Kanakarajan, Bhuvana Kundumani, and Malaikannan Sankarasubbu. 2021. BioELECTRA:pretrained biomedical text encoder using discriminators. In *Workshop on Biomedical Language Processing*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Conglong Li, Minjia Zhang, and Yuxiong He. 2022. The stability-efficiency dilemma: Investigating sequence length warmup for training GPT models. In *NeurIPS*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. 2023. Relora: High-rank training through low-rank updates. *arXiv*.

Sean Lie. 2021. Thinking outside the die: Architecting the ml accelerator of the future.

Sean Lie. 2022a. Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning. *Cerebras Systems Blog*.

Sean Lie. 2022b. Harnessing the Power of Sparsity for Large GPT AI Models. *Cerebras Systems Blog*.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. 2021. Sparse training via boosting pruning plasticity with neuroregeneration.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. 2022a. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *arXiv*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *ACL*.

Valentin Liévin, Christoffer Egeberg Hother, Andreas Geert Motzfeldt, and Ole Winther. 2023. Can large language models reason about medical questions? *arXiv*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv*.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. BioGPT: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*.

Xiaolong Ma, Minghai Qin, Fei Sun, Zejiang Hou, Kun Yuan, Yi Xu, Yanzhi Wang, Yen-Kuang Chen, Rong Jin, and Yuan Xie. 2022. Effective model sparsification by scheduled grow-and-prune methods. In *ICLR*.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv*.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. 2024. ReLU strikes back: Exploiting activation sparsity in large language models. In *ICLR*.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*.

Bethesda (MD) National Library of Medicine. 2003–2023. Pmc open access subset.

NeuralMagic. 2021. Deepsparse.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. 2023a. Capabilities of gpt-4 on medical challenge problems. *arXiv*.

Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolo Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, Renqian Luo, Scott Mayer McKinney, Robert Osazuwa Ness, Hoifung Poon, Tao Qin, Naoto Usuyama, Chris White, and Eric Horvitz. 2023b. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv*.

C.A.I. Peng, Xi Yang, Aokun Chen, Kaleb E. Smith, Nima M. Pournejatian, Anthony B Costa, Cheryl Martin, Mona G. Flores, Ying Zhang, Tanja Magoc, Gloria P. Lipori, Duane A. Mitchell, Naykky Maruquel Singh Ospina, Mustafa Mamon Ahmed, William R. Hogan, Elizabeth A. Shenkman, Yi Guo, Jiang Bian, and Yonghui Wu. 2023. A study of generative large language model for medical research and healthcare. *NPJ Digital Medicine*.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. 2020. What's hidden in a randomly weighted neural network? In *CVPR*.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. *arXiv*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. 2022. Large language models encode clinical knowledge. *arXiv*.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, Nenad Tomasev, Yun Liu, Renee Wong, Christopher Semturs, S. Sara Mahdavi, Joelle Barral, Dale Webster, Greg S. Corrado, Yossi Matias, Shekoofeh Azizi, Alan Karthikesalingam, and Vivek Natarajan. 2023. Towards expert-level medical question answering with large language models. *arXiv*.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 2021. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. *arXiv*.

Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. 2023. Torchsparse++: Efficient training and inference framework for sparse convolution on gpus. In *IEEE/ACM MICRO*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv*.

Vithursan Thangarasa, Abhay Gupta, William Marshall, Tianda Li, Kevin Leong, Dennis DeCoste, Sean Lie, and Shreyas Saxena. 2023a. SPDF: Sparse pre-training and dense fine-tuning for large language models. In *UAI*.

Vithursan Thangarasa, Shreyas Saxena, Abhay Gupta, and Sean Lie. 2023b. Sparse iso-FLOP transformations for maximizing training efficiency. In *NeurIPS Workshop on Advancing Neural Network Training*.

Augustin Toma, Patrick R. Lawler, Jimmy Ba, Rahul G. Krishnan, Barry B. Rubin, and Bo Wang. 2023. Clinical camel: An open expert-level medical language model with dialogue-based knowledge encoding. *arXiv*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal

Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *arXiv*.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Ziheng Wang. 2021. Sparsednn: Fast sparse deep learning inference on cpus. *arXiv*.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023a. Pmc-llama: Towards building open-source language models for medicine. *arXiv*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023b. Bloomberggpt: A large language model for finance. *arXiv*.

Niraj Yagnik, Jay Jhaveri, Vivek Sharma, Gabriel Pila, Asma Ben, and Jingbo Shang. 2024. Medlm: Exploring language models for medical question answering systems. *arXiv*.

Xi Yang, Nima PourNejatian, Hoo Chang Shin, Kaleb E Smith, Christopher Parisien, Colin Compas, Cheryl Martin, Mona G Flores, Ying Zhang, Tanja Magoc, Christopher A Harle, Gloria Lipori, Duane A Mitchell, William R Hogan, Elizabeth A Shenkman, Jiang Bian, and Yonghui Wu. 2022. Gatortron: A large language model for clinical natural language processing. *medRxiv*.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy Liang, and Jure Leskovec. 2022a. Deep bidirectional language-knowledge graph pretraining. In *NeurIPS*.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022b. LinkBERT: Pretraining language models with document links. In *ACL*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *arXiv*.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv*.

A Experimental Setup and Hyperparameter Details

A.1 Pre-training on Biomedical Data

To train all MediSwift models, we use the AdamW optimizer (Loshchilov and Hutter, 2017) with a peak learning rate set at 2×10^{-4}, β_1 =0.9, β_2 =0.95 and ϵ =10^{-8}. A linear warmup period, amounting to 10% of the total training steps, is employed before transitioning to a cosine decay schedule, with the learning rate decreasing to a minimum of 10% of the peak value (i.e., 2×10^{-5}). In Table 5, we provide details on the size and architecture configurations of the MediSwift models we pre-trained. Here, n_{params} is the total number of trainable parameters, n_{layers} is the number of decoder layers, and d_{model} is the base size of the model. The feedforward bottleneck is four times the base size, i.e., $d_{\text{ff}} = 4 \times d_{\text{model}}$. Finally, n_{heads} are the number of attention heads and d_{head} is the dimension of each attention head. The context window size is set to 1024.

Table 5: Sizes, architectures, and pre-training hyperparameters (batch size, learning rate, etc.) of the MediSwift models at three sizes (i.e., Med, Large and XL), which are trained for a total of 104.86B tokens.

	MediSwift Models		
	Med	Large	XL
n_{params}	302M	510M	1.21B
n_{layers}	24	18	24
d_{model}	1024	1536	2048
n_{heads}	16	12	16
d_{head}	64	128	128
Batch Size	512		
MSL	1024		
Optimizer	AdamW		
Warmup Schedule	Linear		
Decay Schedule	Cosine		
LR	2×10^{-4}		
Weight Decay	0.1		
Total Steps	200,000		
Warmup Tokens	10.486×10^{9}		
Training Tokens	104.86×10^{9}		

Following the training FLOPs calculation described in Hoffmann et al. (2022), we compute the total pre-training FLOPs for the dense and sparse variants of MediSwift-Med, Large and XL, and report them in Table 6, along with their relative FLOPs reduction over the dense baseline. Similar to Appendix F of Hoffmann et al. (2022), we also include the training FLOPs contributed by the embedding matrices. Additionally, in large models, the contribution of embedding matrices to the overall FLOPs and parameters is minimal.

Table 6: **Final Pre-training Losses and Computational Efficiency of MediSwift Models.** We summarize the results for the MediSwift-Med and MediSwift-XL models, trained using the biomedical pre-training corpus detailed in Section 3.1. We report the final pre-training losses for dense and sparse variants at 50% and 75% sparsity levels. The total FLOPs and FLOP savings relative to the dense baseline are indicated in parentheses, highlighting the models' computational efficiency.

MediSwift	Size	Pre-train Loss	Train FLOPs (x 10^{20})
$\text{Med}_{\text{dense}}$	302M	2.234	2.677 (1.00x)
$\text{Med}_{s=50\%}$	151M	2.265	1.727 (0.64x)
$\text{Med}_{s=75\%}$	0.76M	2.375	1.252 (0.46x)
$\text{Large}_{\text{dense}}$	510M	2.047	4.248 (1.00x)
$\text{Large}_{s=50\%}$	255M	2.172	2.645 (0.62x)
$\text{Large}_{s=75\%}$	128M	2.281	1.840 (0.43x)
XL_{dense}	1.21B	1.979	9.148 (1.00x)
$\text{XL}_{s=50\%}$	605M	2.012	5.348 (0.58x)
$\text{XL}_{s=75\%}$	302M	2.141	3.448 (0.38x)

A.2 PubMedQA Fine-tuning

As mentioned in Section 4.1, the PubMedQA dataset includes three subsets: PQA-A, PQA-U, and PQA-L. We train all of our MediSwift models on the original train/val/test splits for each of these datasets in a multi-stage manner (Jin et al., 2019), both dense and sparse, using AdamW (Loshchilov and Hutter, 2017) and a linear learning rate warmup (i.e., 10% of to the total training steps) followed by a cosine decay schedule for a maximum 5 epochs, and perform early-stopping when the models began to overfit. We perform a grid search to discover an appropriate learning rate that led to the best downstream classification accuracy on each of the three datasets for a given compute budget. More specifically, on the dense baseline and sparse variants, we select the best batch size among {8, 16, 32, 64} and select the best learning rate among {2e-4, 1e-4, 5e-5, 2.5e-5} on the validation set. After training on the final stage (i.e., PQA-L), we evaluate the model on the PQA-L test set using the official evaluation scripts[6]. All results were averaged across 3 random seeds.

[6] https://github.com/pubmedqa/pubmedqa

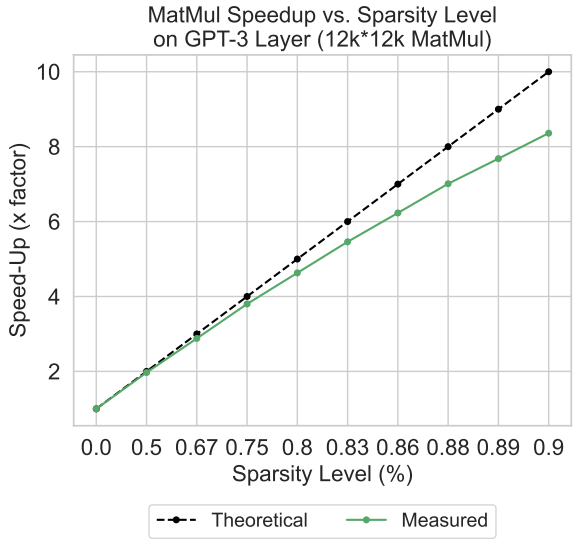

Figure 3: Comparison of measured speedup versus theoretical speedup for GPT-3 layer 12k $\times$ 12k matrix multiplication (MatMul) on the Cerebras CS-2 system at various sparsity levels. This graph illustrates the efficiency gains achieved through sparse computation, highlighting the real-world performance relative to theoretical predictions.

A.3 HoC Fine-tuning

In Section 4.2, we described the Hallmarks of Cancers (HoC) dataset which comprises of 1580 PubMed abstracts with a 1108/157/315 split for train, val and test sets following Gu et al. (2021). We train all of our MediSwift models on the original train/val/test splits for both dense and sparse, using AdamW (Loshchilov and Hutter, 2017) and a linear learning rate warmup (i.e., 10% of to the total training steps) followed by a cosine decay schedule for a total of 100 epochs, and perform early-stopping when the models began to overfit. We perform a grid search to discover an appropriate learning rate that led to the best micro-F1 score for a given compute budget. More specifically, on the dense baseline and sparse variants, we select the best batch size among {16, 32, 64} and select the best learning rate among {8e-5, 4e-5, 2e-5, 1e-5} on the validation set. All results were averaged across 3 random seeds.

B Unstructured Sparsity on Specialized Hardware Accelerators

The Cerebras CS-2 system, designed specifically for accelerating deep learning computations, can handle unstructured sparsity efficiently due to its unique architecture (Lie, 2022b, 2021). The CS-2's wafer-scale engine, with its vast array of computational cores and on-wafer memory, efficiently manages unstructured sparsity's irregular memory access, surpassing traditional architectures that often face memory bandwidth constraints. Moreover, the CS-2 has a significant amount of on-chip memory, reducing the need to access external memory. This is crucial for unstructured sparsity, as the irregular access patterns can lead to high latency if data needs to be fetched from off-chip. By keeping more data on-chip, the CS-2 minimizes these latencies. In addition, the system supports fine-grained parallelism, allowing it to execute many small, sparse operations concurrently across its thousands of cores. This is particularly advantageous for unstructured sparsity, as the workload can be distributed across many cores to maintain high utilization. In Figure 3, we highlight the potential realized gains with unstructured weight sparsity on the Cerebras CS-2.