Pruning as a Domain-specific LLM Extractor

Nan Zhang[♣†] **Yanchi Liu**[♢] **Xujiang Zhao**[♢] **Wei Cheng**[♢]
Runxue Bao[♢] **Rui Zhang**[♣] **Prasenjit Mitra**[♣] **Haifeng Chen**[♢]
[♣]The Pennsylvania State University [♢]NEC Labs America
{njz5124,rmz5227,pmitra}@psu.edu
{yanchi,xuzhao,weicheng,rbao,haifeng}@nec-labs.com

Abstract

Large Language Models (LLMs) have exhibited remarkable proficiency across a wide array of NLP tasks. However, the escalation in model size also engenders substantial deployment costs. While few efforts have explored model pruning techniques to reduce the size of LLMs, they mainly center on general or task-specific weights. This leads to suboptimal performance due to lacking *specificity* on the target domain or *generality* on different tasks when applied to domain-specific challenges. This work introduces an innovative unstructured dual-pruning methodology, D-PRUNER, for domain-specific compression on LLM. It extracts a compressed, domain-specific, and task-agnostic LLM by identifying LLM weights that are pivotal for general capabilities, like linguistic capability and multi-task solving, and domain-specific knowledge. More specifically, we first assess general weight importance by quantifying the error incurred upon their removal with the help of an open-domain calibration dataset. Then, we utilize this general weight importance to refine the training loss, so that it preserves generality when fitting into a specific domain. Moreover, by efficiently approximating weight importance with the refined training loss on a domain-specific calibration dataset, we obtain a pruned model emphasizing *generality* and *specificity*. Our comprehensive experiments across various tasks in healthcare and legal domains show the effectiveness of D-PRUNER in domain-specific compression. Our code is available at https://github.com/psunlpgroup/D-Pruner.

1 Introduction

Large Language Models (LLMs) such as the GPT family (Brown et al., 2020) and the LLaMA family (Touvron et al., 2023) have exhibited remarkable advancements across a diverse spectrum of NLP tasks. However, the substantial size of LLMs engenders cost-intensive deployment in real-world applications and renders them unsuitable for scenarios necessitating efficient inference and low latency (Bai et al., 2024). Recently, model pruning techniques have been successfully applied to language models (Han et al., 2015; Xia et al., 2022; Frantar and Alistarh, 2023). These methods aim to yield a compact language model characterized by a significantly reduced parameter count, which is cost-efficient for deployment. However, most of them target relatively small language models, and only a few focus on LLMs (Frantar and Alistarh, 2023; Ma et al., 2023; Sun et al., 2023; Xia et al., 2023). Moreover, the existing strategies mainly center on general or task-specific weights, leading to suboptimal performance due to lacking *specificity* on the target domain or *generality* on different tasks when applied to domain-specific challenges. Here ***generality*** refers to the general capabilities of an LLM such as language understanding and generation, and multi-task solving, and ***specificity*** refers to the capability of an LLM to understand domain-specific knowledge.

As shown in Figure 1, the weights in an LLM work together to support its general capabilities and to store various domain knowledge. The domain-shared weights (or general weights) empower the LLM with linguistic and multi-task solving prowess akin to human language usage and thinking. The domain-specific weights (or domain weights) are pivotal for endowing the LLM with domain-specific expertise mirroring that of domain experts. However, the current pruning methods mainly focus on preserving general or task-specific weights, which may not be enough to deal with domain-specific problems. For example, post-training pruning methods (Frantar and Alistarh, 2023) assume the model is optimized and prune unimportant weights based on an open-domain calibration dataset. This leads to a pruned model that

[†]Work done as a Research Intern at NEC Labs America.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 1417–1428

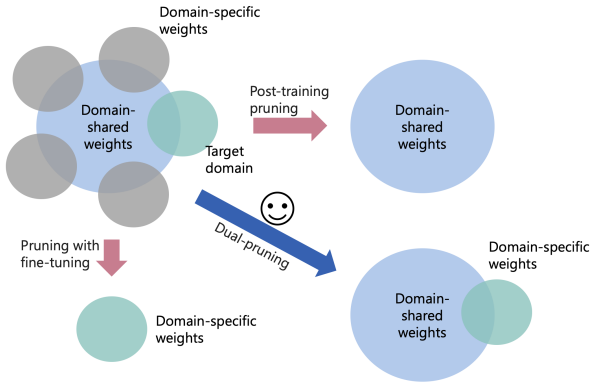


Figure 1: Different types of pruning methods. An LLM is composed of domain-shared weights and domain-specific weights. Post-training pruning focuses on domain-shared weights for generality, pruning with fine-tuning focuses on domain-specific weights for specificity, and our dual-pruning method preserves weights pivotal for both generality and specificity.

focuses on model generality with domain-specific weights not considered. On the other hand, pruning with fine-tuning methods (Ma et al., 2023) utilizes gradients during fine-tuning on a specific task to estimate the importance of parameters. As a result, the pruned model focuses on the model specificity while decreasing the linguistic and multi-task solving capabilities, compromising the LLM's capacity as a versatile task-agnostic solver.

To this end, this study introduces a novel dual-pruning approach, D-PRUNER, for domain-specific unstructured pruning on LLMs, which aims to extract a domain-specific LLM from the foundation LLM. This extracted model is able to solve different tasks in the target domain and facilitates further domain-specific fine-tuning. D-PRUNER is designed to harness calibration data for guiding LLM pruning processes while preserving generality and specificity for multi-task solving and domain challenges. The resulting compressed LLM can be seamlessly adapted to the target domain, enabling deployment with limited computing resources. Specifically, D-PRUNER adeptly captures and retains both general and domain parameters while selectively eliminating insignificant model parameters. This mechanism comprises the following steps: firstly, a general weight importance module operates to assess the significance of model parameters for general capabilities. Subsequently, we propose an updated training loss function based on the autoregressive training objective for the next token prediction by integrating the general importance as a regularization term. This way, we identify weights contributing to both generality and domain specificity when training on a domain calibration dataset. Then, with the updated loss function, we compute the weight importance leveraging gradients without updating the model. Moreover, an approximation algorithm, empirical Fisher (Martens, 2020; Sung et al., 2021), is utilized to compute the weight importance efficiently for pruning.

We evaluate the performance of D-PRUNER on LLaMA2 (Touvron et al., 2023), a widely adopted open-source LLM. Our experimental findings demonstrate that D-PRUNER exhibits remarkable efficiency in the extraction of sparse domain networks from pre-trained LLMs, with a limited amount of calibration data provided. Remarkably, D-PRUNER achieves comparable results to the full dense model while achieving 50% sparsity, surpassing the performance of alternative pruning techniques across diverse domain-specific datasets in healthcare and legal domains encompassing language comprehension, question answering, and summarization tasks.

2 Related Work

Model compression involves transforming a large, resource-intensive model into a compact version suitable for low-resource deployment (Deng et al., 2020; Zhu et al., 2023). There are mainly three techniques for model compression, which are pruning, knowledge distillation, and quantization.

Pruning. Pruning techniques in neural networks can be broadly classified into structured pruning and unstructured pruning (Xia et al., 2022; Sanh et al., 2020; Du et al., 2021). *Structured pruning* entails the removal of entire network components, such as channels or layers, guided by specific criteria, while maintaining the overall network architecture. In contrast, *unstructured pruning* targets individual weights, leading to an irregular sparse structure.

While numerous attempts have been made to prune language models of relatively small scales, such as BERT (Kenton and Toutanova, 2019), scant attention has been devoted to pruning LLMs containing billions of parameters. These larger models possess 100-1000 times more weights, rendering the pruning task significantly more challenging. SparseGPT (Frantar and Alistarh, 2023), a post-training method for Large Language Models (LLMs), lacks the capability to identify crucial

weights tailored to specific domains or tasks as it refrains from fine-tuning. On the other hand, LLM-Pruner (Ma et al., 2023) employs gradient-based techniques for pruning. However, it falls short in identifying pivotal weights essential for domain-shared knowledge, resulting in pruned models that lack the desired level of generality.

The existing pruning methods either focus on general or domain-specific weights, yet none of them consider preserving both at the same time. To the best of our knowledge, we are the first to work on pruning LLMs while preserving weights important to both generality and specificity.

Knowledge Distillation. Knowledge Distillation (KD) has emerged as a powerful technique, drawing considerable interest for its ability to augment model performance and enhance generalization capacities (Hinton et al., 2015; Zhu et al., 2023). At its core, KD revolves around the transfer of expertise from a complex model, referred to as the "teacher model", to a simplified counterpart known as the "student model". This intricate process of knowledge transfer aims to distill the profound insights encapsulated within the teacher models, condensing them into a more concise and efficient representation within the student models.

While KD has been proven a powerful tool for model compression, it needs specific downstream tasks and a large amount of data for the student models to learn from the teacher models. Thus, the output that student models produce mainly focuses on a specific task and loses the generality capability. KD generally sets higher requirements on data availability and computation budgets (e.g., GPU memory) than pruning.

Quantization. In the realm of model compression, quantization has emerged as a widely embraced technique to alleviate the storage and computational challenges inherent in deep learning models (Guo et al., 2020; Dettmers et al., 2021, 2022, 2023). Conventional model representations rely on floating-point numbers, but quantization converts them into integers or discrete forms. This transformation leads to substantial reductions in storage requirements and computational complexities. While a certain degree of precision loss is inevitable, carefully designed quantization methods can achieve significant model compression with minimal accuracy degradation. Although challenges remain, such as maintaining model interpretability and addressing task-specific intricacies, the current body of research establishes a robust groundwork for ongoing advancements in LLM quantization, which could be complementary to LLM pruning.

3 Methodology

To preserve both generality and specificity on the pruned model, our dual-pruning method D-PRUNER considers weights important to both generality and specificity during training on a calibration dataset. Note we only use the weight gradient generated from the training process but do not update the model weights. Our model is pruned in a task-agnostic fashion (e.g., we adopted a pre-training objective, next token prediction, as a part of training loss) so that the pruned model can solve different tasks in the target domain.

D-PRUNER comprises the following steps: firstly, a general weight locating module operates to assess the significance of model parameters for general understanding (Section 3.1). Subsequently, an updated loss function for the training process is proposed by integrating the general weight importance as a regularization term. This way, we identify weights contributing to both general and domain knowledge (Section 3.2). Finally, with the updated loss function, we compute the weight gradients on a small domain calibration dataset without updating the model and approximate our dual-pruning weight importance by utilizing the empirical Fisher index (Sung et al., 2021) for pruning (Section 3.3).

Our method concentrates on unstructured pruning in a layer-by-layer manner for the Transformers model. We consider query, key, value, and output projections of all self-attention layers and gate (Liu et al., 2021), down, and up projections of all MLP (multilayer perceptron) layers for pruning.

3.1 General Weight Importance

The first step of our method involves locating important weights in terms of general knowledge. Following the same hypothesis as Frantar and Alistarh (2023), we assume that an important weight will cause a larger increase in loss value than those less important ones if it is pruned (set to 0) during training. Formally, if a dataset of the open-domain calibration $\mathcal{D}_g = \{x_j, y_j\}_{j=1}^{N}$ with size N is used for training and W stands for weight matrices of a model, the importance of each weight at index m, denoted as I_{W^m}, can be approximated using Taylor

series as shown by LeCun et al. (1989):

$$
\begin{aligned}
I_{W^m} &= |\mathcal{L}(\mathcal{D}_g) - \mathcal{L}_{W^m=0}(\mathcal{D}_g)| \\
&= |\frac{\partial \mathcal{L}(\mathcal{D}_g)}{\partial W^m} W^m + \frac{1}{2} W^m H_{mm} W^m \\
&+ O(||W^m||^3)|
\end{aligned} \tag{1}
$$

where H denotes the Hessian matrix, and $\mathcal{L}$ is the cross-entropy loss. For a model that is sufficiently trained to a local minimum on its loss curvature (e.g., pretrained foundational language models such as LLaMA), the classic Optimal Brain Surgeon (Hassibi et al., 1993) further approximates the importance of W^m as:

$$
\varepsilon^m = \frac{1}{2} \frac{(W^m)^2}{[H^{-1}]_{mm}} \tag{2}
$$

ε^m can also be viewed as the error caused by removing the weight W^m. We compute ε^m for all the weights subject to pruning and construct a matrix of importance scores G with respect to general domains that have the same dimension as W.

3.2 Updated Loss with Regularization

To identify the weights that are important in both general and domain-specific knowledge, we modify the original loss function of LLM training. In LLM training, cross-entropy loss is used in the next token prediction task (Radford et al., 2018). Similar to Thompson et al. (2019), we add a regularization term to constrain the change of important general weights found in the first step. Suppose that there are M number of prunable weights in total. To train on a domain-specific calibration dataset $\mathcal{D}_s = \{x_j, y_j\}_{j=1}^{P}$, we add the proposed regularization term on top of the next token prediction loss $\mathcal{L}_{next}$ to obtain our final training objective:

$$
\mathcal{L}_{\text{ours}} = \mathcal{L}_{\text{next}} + \lambda \sum_{m=1}^{M} G^m (W^{m\prime} - W^m)^2 \tag{3}
$$

where G^m is the general weight importance, $W^{m\prime}$ denotes the updated weight value of W^m, λ is a hyperparameter, and the second term on the right is $\mathcal{L}_{\text{regular}}$.

In practice, the direct calculation of this regularization term in the forward pass is computationally expensive for two reasons: (1) it involves both W^m and G^m which are very large, and (2) gathering updated model parameters ($W^{m\prime}$) in a partitioned (Rasley et al., 2020) or sharded (Zhao et al., 2023) system is inefficient. Based on the recent success of applying gradient descent on full fine-tuning of LLMs (Lv et al., 2023), we choose to use gradient descent to optimize parameters. Therefore, at a learning rate α, denoting the gradient of each parameter with respect to $\mathcal{L}_{\text{next}}$ as g^m_{next}, we reduce the regularization term to:

$$
\begin{aligned}
\mathcal{L}_{\text{regular}} &= \sum_{m=1}^{M} G^m (W^{m\prime} - W^m)^2 \\
&= \lambda \sum_{m=1}^{M} G^m (W^m - \alpha g^m_{\text{next}} - W^m)^2 \\
&= \lambda \sum_{m=1}^{M} \alpha^2 G^m (g^m_{\text{next}})^2
\end{aligned} \tag{4}
$$

During the backward pass, optimizing this regularization term requires second-order derivatives, which indicates that Hessian matrices (H) are needed. Directly computing the Hessian matrices is infeasible for such a large number of parameters. Therefore, we use the Fisher information matrix to approximate the diagonal of the Hessian (Sung et al., 2021). And the Fisher information matrix can be further approximated by the average of the squared gradient of the model's prediction over P. We write the gradient of the regularization with respect to every parameter matrix in a finer granularity:

$$
\frac{\partial \mathcal{L}_{\text{regular}}}{\partial W^m} \approx 2\lambda\alpha^2 G^m g^m_{\text{next}} H_{mm} \tag{5}
$$

$$
H_{mm} \approx \frac{1}{P} \sum_{j=1}^{P} (g^m_{\text{next}}(x_j, y_j))^2 \tag{6}
$$

We directly compute $\frac{\partial \mathcal{L}_{\text{regular}}}{\partial W}$ via Equation 5 above instead of relying on PyTorch backward pass to maximize computing efficiency. The final gradient computation of our regularized loss function is shown below:

$$
\frac{\partial \mathcal{L}_{\text{ours}}}{\partial W^m} = \frac{\partial \mathcal{L}_{\text{next}}}{\partial W^m} + \frac{\partial \mathcal{L}_{\text{regular}}}{\partial W^m} \tag{7}
$$

3.3 Dual-pruning Importance Score

Finally, we calculate the dual-pruning importance score of each weight, and unimportant weights can be pruned according to their importance. We use Equation 1 for importance estimation instead of Equation 2, because our model has not converged to an optimum on the target domain. However, direct computation of the Hessian matrix in Equation 2 is

	InternalMed_Harrison	MedNLI	PubMedQA	HQS	MultiLegalPile	CaseHOLD	BillSum
Domain	Healthcare	Healthcare	Healthcare	Healthcare	Legal	Legal	Legal
Task / Type	Generation	NLI	QA	Summarization	Generation	QA	Summarization
# Instances in Test	300	1422	500	100	300	200	200
Metrics	Perplexity	Accuracy	Macro-F1	ROUGE	Perplexity	Macro-F1	ROUGE

Table 1: Details of each dataset that we use for model evaluation.

infeasible since it involves $O(M^2)$ complexity for each weight update. Therefore, we also leverage Sung et al. (2021) to approximate the diagonal of the Hessian, and the final importance score S^m can be defined as:

$$S^m \approx |\frac{\partial \mathcal{L}_{\text{ours}}(\mathcal{D}_s)}{\partial W^m} W^m + \frac{1}{2}[\frac{\partial \mathcal{L}_{\text{ours}}(\mathcal{D}_s)}{\partial W^m} W^m]^2 + O(||W^m||^3)| \tag{8}$$

Here $O(||W^m||^3)$ can be neglected according to the quadratic approximation (LeCun et al., 1989). Note the calculation of S^m considers both general and domain-specific knowledge via our regularized training objective. Combining both regularization and importance estimation via empirical Fisher approximation, our method expects to conduct pruning that maintains weights important to both general and domain-specific knowledge, thus preserving generality and specificity. And these importance scores are used to guide our pruning decisions. For example, if we set the sparsity level to be 50%, weights that have the smallest 50% of importance scores in each layer will be pruned.

4 Experiment Setup

We evaluate D-PRUNER on two knowledge-intensive domains, which are healthcare and legal. For model generality under domain-specific challenges, we evaluate the linguistic capability using domain text generation, and evaluate the multi-task solving capability on different domain tasks, i.e., natural language inference (NLI), question answering (QA), and summarization. Since we use domain datasets, the model specificity on domains can also be evaluated. In addition, we fine-tune the pruned model on domain datasets to further evaluate the generality and specificity.

We evaluate D-PRUNER on the LLaMA2 model family, which is the most used open-source LLM. We mainly apply our pruning method and baseline methods to LLaMA2-7B and LLaMA2-13B to show our results. Our method can also be easily applied to other LLMs with different sizes and architectures. For instance, Appendix B shows further experiment on BLOOM model (Le Scao et al., 2022).

4.1 Iterative blocking

Motivated by Frantar and Alistarh (2023), we perform experiments (in Table 2) on D-PRUNER with and without iterative blocking. Iterative blocking means to make pruning decisions for every fixed number (B_s) of columns within a weight matrix. In other words, instead of selecting a single pruning mask for an entire weight matrix, a pruning sub-mask is selected for every B_s columns to reach overall sparsity level. We set $B_s = 128$ for weight matrices with the smallest number of columns and increase B_s for those with more columns. Except Table 2, D-PRUNER in other tables does not adopt iterative blocking.

4.2 Datasets and Evaluations

Datasets. Table 1 shows the details of each dataset that we used. Specifically, for healthcare, we select a medical textbook InternalMed_Harrison (Bigby, 1988), MedNLI (Romanov and Shivade, 2018), PubMedQA (Jin et al., 2019), and Health Question Summarization (HQS) from the MEDIQA 2021 shared task 1 (Ben Abacha et al., 2021; Ben Abacha and Demner-Fushman, 2019) as domain datasets. For legal domain, we select MultiLegalPile (Niklaus et al., 2023), CaseHOLD (Zheng et al., 2021), and BillSum (Kornilova and Eidelman, 2019). As for open-domain calibration data, we extract text from C4 dataset (Raffel et al., 2019).

To construct our domain-specific calibration data, we select training instances from MedNLI, PubMedQA, and HQS at a ratio of 20%/60%/20% and from CaseHOLD and BillSum at a ratio of 50%/50%. These ratios are determined based on the difficulties and training sizes of these benchmarks. Both NLI and QA tasks that we adopt are asking models to perform classification. We experiment with different sizes of the domain-specific calibration dataset and find a size of 1000 achieves the best trade-off in terms of pruning efficiency and

effectiveness for both domains. For model evaluation, besides using the test instances of those benchmarks, we leverage InternalMed_Harrison and MultiLegalPile for perplexity evaluation. 300 paragraphs are selected from each data source to form the test set of perplexity. Note that we use a subset of all the test examples of CaseHOLD and BillSum, since these two benchmarks are significantly larger in size and their individual instance tends to be longer.

Evaluation Metrics. We first evaluate the linguistic capability of pruned models on InternalMed_Harrison and MultiLegalPile using perplexity. We then evaluate the multi-task solving capability and domain specificity on different domain tasks. Specifically, we choose accuracy metric for NLI task (MedNLI), macro-F1 for QA tasks (PubMedQA and CaseHOLD), and ROUGE scores (Lin, 2004) for summarization tasks (HQS and BillSum).

4.3 Baselines

We compare our method with a variety of LLM pruning baselines. All methods are applied to the same foundation model (either 7B of 13B of LLaMA2) for fair comparisons. As an ablation study, we also evaluate an unstructured pruning method using weight gradient by removing the regularization term in the training loss of D-PRUNER.

- **Magnitude pruning** prunes weights based on their magnitudes (Han et al., 2015). We follow the standard practice of magnitude pruning on language models, where weights are compared layer-wise. Magnitude pruning is a simple and robust baseline that has been demonstrated to outperform many other pruning methods.
- **LLM-Pruner** is a structured pruning method using weight gradient to evaluate weight importance (Ma et al., 2023). A calibration dataset is used for its gradient calculation, so we combine both open-domain (C4) and domain-specific calibration data when we use LLM-Pruner.
- **SparseGPT** is an unstructured post-training pruning method (Frantar and Alistarh, 2023). It uses an efficient weight update procedure that iterates between weight removal and weight update at each layer. It also uses a calibration dataset for approximation. Thus, similarly to D-PRUNER and LLM-Pruner, we use open-domain and domain-specific calibration data for fair comparisons.

Moreover, for all the baseline methods, we continue to fine-tune their pruned models using LoRA (Hu et al., 2021) on all the datasets together (NLI, QA, and summarization data combined) in each domain and then test the fine-tuned model on the datasets in Table 1. We only use the default open-domain calibration dataset for the pruned models of LLM-Pruner and SparseGPT at this step, because these models will eventually undergo LoRA fine-tuning. Data instances of our fine-tuning dataset follow the Alpaca (Taori et al., 2023) template so that models are trained to predict the responses. Specifically, for healthcare, we have 7000, 7000, and 1000 training instances from MedNLI, PubMedQA, and HQS, respectively. For legal domain, we have 13000 training instances from CaseHOLD and 2000 from BillSum.

4.4 Implementation Details

We perform prompt engineering in a zero-shot setting before prompting a series of models. The finalized prompt is kept the same across all candidate models on one task to ensure fairness. The hyperparameters used by different models are in Appendix C.

5 Results and Analysis

Our results and analysis aim to answer the following research questions:

- RQ 1: How does D-PRUNER compare against other pruning baselines (5.1)?
- RQ 2: What are the performance of all candidate models after LoRA fine-tuning (5.2)?
- RQ 3: As an important contribution of D-PRUNER, is dual-pruning an effective method of compressing LLM (5.1, 5.3, and 5.5)?
- RQ 4: How does D-PRUNER perform under different sparsity levels or different sizes of domain-specific calibration data (5.4)?

5.1 Overall Results

Our overall results for the two domains are presented in Table 2. All models are pruned to 50% sparsity level except the dense one.

Improvement on NLI and QA D-PRUNER delivers consistent score improvement on NLI and QA tasks when it is compared against baselines based on LLaMA2-7B and LLaMA2-13B. With two exceptions, variants of D-PRUNER based on the inclusion and exclusion of iterative blocking outperform baselines on 4 out of 6 cases when classification is performed (MedNLI, PubMedQA,

Model	Healthcare						Legal				
	Perplexity	MedNLI	PubMedQA	R1	R2	RL	Perplexity	CaseHOLD	R1	R2	RL
LLaMA2-7B											
Dense	5.49	37.62	23.77	22.51	7.18	19.50	2.26	28.82	32.64	18.32	26.48
Magnitude (Han et al., 2015)	16.08	33.90	28.29	9.60	1.63	8.09	8.64	23.84	7.84	2.21	6.13
LLM-Pruner (Ma et al., 2023)	88.25	33.90	22.34	5.52	0.30	5.45	32.22	13.59	6.76	0.72	5.40
SparseGPT (Frantar and Alistarh, 2023)	**6.39**	33.47	36.22	22.60	7.68	19.13	**2.62**	28.41	32.68	**18.89**	26.19
D-PRUNER (w/ iterative blocking)	7.07	34.53	**45.38**	24.72	8.87	21.09	2.70	**30.56**	**33.77**	18.53	**26.25**
D-PRUNER (w/o iterative blocking)	6.96	**34.81**	42.40	**25.05**	**9.65**	**22.34**	2.72	26.14	32.14	18.42	26.14
LLaMA2-13B											
Dense	5.20	35.02	40.54	19.26	5.80	16.40	2.12	28.89	35.34	21.19	27.82
Magnitude (Han et al., 2015)	6.59	**36.71**	45.12	19.60	5.01	16.33	2.81	21.95	29.90	16.94	24.51
LLM-Pruner (Ma et al., 2023)	23.95	34.39	17.37	7.60	1.24	7.00	12.16	13.46	17.21	3.08	12.37
SparseGPT (Frantar and Alistarh, 2023)	**5.77**	34.39	52.65	22.25	**8.35**	19.19	**2.39**	**28.62**	33.68	19.35	27.60
D-PRUNER (w/ iterative blocking)	6.30	34.88	**52.86**	20.56	6.95	17.85	2.40	28.30	33.83	20.51	27.56
D-PRUNER (w/o iterative blocking)	6.16	35.16	50.87	**23.99**	7.78	**20.04**	2.40	27.27	**35.77**	**21.81**	**28.42**

Table 2: Overall results when candidate models (at 50% sparsity) are tested on two domains. The best scores are in **bold** except the ones from the dense models. Note that the ROUGE scores reported in the healthcare domain correspond to HQS dataset while those in the legal domain correspond to BillSum. Perplexity in healthcare is tested on InternalMed_Harrison and perplexity in legal is tested on MultiLegalPile.

Model (Fine-tuned with LoRA)	Healthcare						Legal				
	Perplexity	MedNLI	PubMedQA	R1	R2	RL	Perplexity	CaseHOLD	R1	R2	RL
LLaMA2-7B											
Dense	5.68	64.84	41.37	33.26	12.60	28.92	2.26	28.82	34.64	20.47	28.33
Magnitude (Han et al., 2015)	8.39	62.59	23.71	32.02	12.25	29.27	7.28	25.89	17.64	8.19	14.52
LLM-Pruner (Ma et al., 2023)	44.56	58.72	26.78	22.21	6.12	20.57	215.13	14.37	7.97	0.78	6.68
SparseGPT (Frantar and Alistarh, 2023)	**6.44**	**68.85**	27.37	28.97	11.27	25.93	2.86	27.31	27.79	17.55	23.74
D-PRUNER	6.74	61.88	**32.58**	**36.49**	**13.71**	**31.85**	**2.73**	**27.58**	**31.00**	**19.03**	**25.96**

Table 3: Results of fine-tuned candidates models at 50% sparsity. LoRA fine-tuning is conducted on D-PRUNER without iterative blocking.

and CaseHOLD on both 7B and 13B LLaMA2) in Table 2. It is clear to see that magnitude pruning and SparseGPT are generally stronger models than LLM-Pruner. The dense model sometimes has worse scores than others across 7B and 13B LLaMA2, which indicates that scaling parameters of a pre-trained language model does not necessarily increase the performance on a single benchmark on NLI and QA. We can see that iterative blocking generally yields better scores on these classification tasks such as reaching 30.56 F1 score on CaseHOLD based on LLaMA2-7B, which is a significant improvement over baselines and D-PRUNER without it. Thus, we recommend to adopt iterative blocking on the classification tasks when strong domain knowledge is required.

Improvement on Summarization D-PRUNER presents the strongest summarization performance. The most exciting thing is that its ROUGE scores are mostly higher than the dense ones. We notice the top summarization performance of LLaMA2-13B-based models on HQS is lower than that of LLaMA2-7B-based models, which is counterintuitive. According to the state-of-the-art of HQS (Zhang et al., 2023; He et al., 2021), we find that D-PRUNER is close to the best ROUGE scores produced by single systems, so we consider that this dataset is relatively simple. Thus, our LLaMA2-7B-based models seem to find an upper limit of ROUGE given the existing reference summaries, so going from 7B to 13B incurs a small performance degradation on dense model, SparseGPT, and D-PRUNER. The strong summarization performance of D-PRUNER on both domains demonstrates its usability as an efficient and domain-specific language model. As for iterative blocking, D-PRUNER without it generally has better perplexity and summarization performance. However, considering the exception in the legal domain based on LLaMA2-7B, we recommend to check perplexity scores on the validation data when deciding whether to use iterative blocking for perplexity and summarization assessment.

Improvement on Perplexity D-PRUNER has the second best perplexity scores on healthcare and legal domains across 7B and 13B LLaMA2.

These scores reflect the strong linguistic capabilities of SparseGPT and D-PRUNER when they encounter knowledge-intensive domains. D-PRUNER does not surpass SparseGPT on perplexity metric, and the reason might come from the fine-tuning pipeline (Lv et al., 2023) we use. Lv et al. (2023) is a full-parameter fine-tuning pipeline that aims towards GPU memory efficiency, so its effectiveness on a specific metric might be compromised. Moreover, we suspect that the data we use from InternalMed_Harrison and MultiLegalPile may be closer to the general domain both semantically and syntactically. Since SparseGPT prunes LLM mainly based on generality, it has better perplexity scores than ours.

5.2 Performance After Fine-tuning

Table 3 shows the results of fine-tuned candidate models at 50% sparsity. Similar to the performance discussed above, D-PRUNER always delivers the best summarization scores and mostly presents the best classification results after fine-tuning, which demonstrates that fine-tuning can further improve the pruning performance of our method. For most models, macro-F1 on PubMedQA decreases after fine-tuning, because this test set is imbalanced and models mostly learn to predict the majority class labels. In fact, the accuracies of most models on PubMedQA increase after fine-tuning as shown in Appendix A, so this fine-tuning method still makes a difference. We also do not see too much score improvement for many models on CaseHOLD, since it is a quite challenging task for our experiment setting (e.g., we combine only a small subset of original training data for each task and perform multi-task fine-tuning as discussed in Section 4).

5.3 Ablation Study

In Table 4, we show that pruning without integrating general domain importance as a regularization term yields suboptimal performance. In other words, this means to remove the consideration of generality. We find perplexities in both domains are higher than pruning with regularization. This demonstrates that our dual pruning mechanism that considers both generality and specificity is able to improve model performance.

5.4 Effect of Sparsity and Domain Calibration Data

In Table 5, it is clear that perplexity keeps increasing when D-PRUNER becomes more sparse, which is expected. Since 50% sparsity is a good balance between sparsity and performance, we select it to report our performance in Table 2 and 3.

Model	Healthcare perplexity	Legal perplexity
no regularization	7.23	2.82
D-PRUNER	**6.96**	**2.72**

Table 4: Results of removing the regularization.

Sparsity	Healthcare perplexity	Legal perplexity
10%	5.49	2.26
20%	5.52	2.27
30%	5.61	2.31
40%	5.91	2.42
50%	6.96	2.72
60%	15.19	4.59
70%	223.63	84.25

Table 5: Results of changing sparsities on D-PRUNER.

# samples	Healthcare perplexity	Legal perplexity
100	8.18	3.34
500	7.15	2.97
1000	6.96	2.72
1500	7.96	2.70

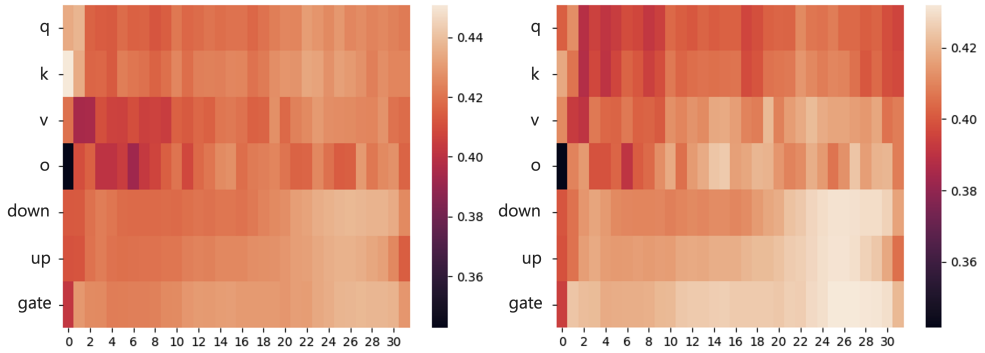
Table 6: Results of trying different sizes of domain-specific calibration data.

Based on Table 6, we believe setting the size of domain-specific calibration data to 1000 is reasonable. As the last row shows, increasing its size does not always guarantee a performance improvement.

5.5 Mask Similarity

To better understand the pruned model on different domains, we compare the similarity of the pruning masks. In our study on LLaMA2-7B, each generated mask contains 7*32 matrices for 32 layers and 7 projection matrices in the self-attention module (q, k, v, o) and MLP module (down, up, gate) in each layer. For each matrix, we calculate the similarity as the number of shared "1" elements ("1" means weights not pruned) in the two masks divided by the matrix size. Note all the masks are generated in 50% sparsity.

Figure 2 (a) shows the mask similarity between the open-domain and healthcare domain, and 2 (b) shows the mask similarity between the healthcare domain and legal domain. The results show that the masks are quite different, with shared elements as low as 35%. Generally, the self-attention modules



(a) Open-domain vs healthcare domain.

(b) Healthcare domain vs legal domain.

Figure 2: Illustration of mask similarity. It shows that masks for different domains are quite different. The self-attention modules contribute more to specificity, and MLP modules store knowledge that is shared by different domains.

share fewer elements than the MLP modules. This means self-attention modules contribute more to specificity, and MLP modules store knowledge that is shared by different domains.

6 Conclusion

We introduce D-PRUNER, an innovative unstructured dual-pruning method for domain-specific compression on LLM. It is able to extract a compressed, domain-specific, and task-agnostic LLM by identifying weights that are pivotal for both generality and specificity. More specifically, the general weight importance is first assessed by quantifying the error incurred upon their removal with the help of open-domain calibration data. Then, we utilize this general weight importance to refine our training loss, so that it considers generality when fitting into a specific domain. Moreover, by efficiently approximating weight importance with the refined training loss on a domain-specific calibration dataset, we obtain a pruned model emphasizing general capabilities and domain-specific knowledge. Our comprehensive experiments across various tasks in different domains show the effectiveness of D-PRUNER in domain-specific pruning.

Limitations

Although D-PRUNER presents strong performance in Section 5, many of its perplexity scores reach the second place in healthcare and legal domains (dense model is not counted here). Further improving this perplexity is a valuable extension of this paper.

Another limitation of this work is that D-PRUNER is more memory-intensive than SparseGPT during pruning, since D-PRUNER is based on full-parameter fine-tuning and SparseGPT does not leverage global gradient information. D-PRUNER sets similar memory requirement as LLM-Pruner. As a trade-off, D-PRUNER reaches better performance on most of the metrics. It is also more flexible, since it computes matrices of importance scores without actually sparsifying LLMs. Therefore, researchers can make real-time decisions about the desired sparsity level, and changing the sparsity is very efficient.

Acknowledgments

We thank Yusen Zhang, Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Xiaoxin Lu, and Ryo Kamoi for the valuable discussions and comments. We also would like to thank the anonymous reviewers for their helpful comments.

References

Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. 2024. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*.

Asma Ben Abacha and Dina Demner-Fushman. 2019. On the summarization of consumer health questions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2228–2234, Florence, Italy. Association for Computational Linguistics.

Asma Ben Abacha, Yassine Mrabet, Yuhao Zhang, Chaitanya Shivade, Curtis Langlotz, and Dina Demner-Fushman. 2021. Overview of the MEDIQA 2021 shared task on summarization in the medical domain. In *Proceedings of the 20th Workshop on Biomedical*

Language Processing, pages 74–85, Online. Association for Computational Linguistics.

JudyAnn Bigby. 1988. Harrison's principles of internal medicine. *Archives of Dermatology*, 124(2):287–287.

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. *Advances in neural information processing systems*, 33:1877–1901.

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Mengnan Du, Subhabrata Mukherjee, Yu Cheng, Milad Shokouhi, Xia Hu, and Ahmed Hassan Awadallah. 2021. Robustness challenges in model distillation and pruning for natural language understanding. *arXiv preprint arXiv:2110.08419*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pages 3887–3896. PMLR.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Yifan He, Mosha Chen, and Songfang Huang. 2021. damo_nlp at MEDIQA 2021: Knowledge-based preprocessing and coverage-oriented reranking for medical question summarization. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 112–118, Online. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. 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 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Anastassia Kornilova and Vladimir Eidelman. 2019. BillSum: A corpus for automatic summarization of US legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56, Hong Kong, China. Association for Computational Linguistics.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. 2023. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.

James Martens. 2020. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851.

Joel Niklaus, Veton Matoshi, Matthias Stürmer, Ilias Chalkidis, and Daniel E. Ho. 2023. Multilegalpile: A 689gb multilingual legal corpus.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, Brussels, Belgium. Association for Computational Linguistics.

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

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

Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2062–2068.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 1513–1528.

Nan Zhang, Yusen Zhang, Wu Guo, Prasenjit Mitra, and Rui Zhang. 2023. FaMeSumm: Investigating and improving faithfulness of medical summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10915–10931, Singapore. Association for Computational Linguistics.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. 2021. When does pretraining help? assessing self-supervised learning for law and the casehold dataset. In *Proceedings of the 18th International Conference on Artificial Intelligence and Law*. Association for Computing Machinery.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*.

A Accuracy Scores on PubMedQA

In Table 7, we report the accuracy score of each model on PubMedQA before and after LoRA fine-tuning. Except LLM-Pruner, we see score improvement on all other models after fine-tuning. Thus, Table 7 indicates that our fine-tuning is still improving model performance on PubMedQA in some ways.

B Experiments on BLOOM

We conduct a small set of experiments in healthcare domain for illustrative purpose. SparseGPT is chosen for comparison, since it is the strongest baseline. We run SparseGPT under two settings: (1) only open-domain calibration dataset is used for pruning, and (2) both open-domain and domain-specific calibration datasets are used, which is the same as the setting in Section 5. All BLOOM experiments are based on the `bigscience/bloom-7b1` model on Hugging Face.

As shown in Table 8, SparseGPT yields the best performance on BLOOM across all three metrics. Although D-PRUNER surpasses SparseGPT

Model	Before LoRA	After LoRA
Dense	39.20	64.60
Magnitude	47.00	55.20
LLM-Pruner	51.40	40.20
SparseGPT	53.80	57.00
D-PRUNER	58.80	59.20

Table 7: Accuracy scores of different models on PubMedQA dataset.

Model	Perplexity	MedNLI	PubMedQA
Dense	9.40	33.26	23.72
SparseGPT*	11.16	32.07	29.74
SparseGPT	10.88	33.47	24.23
D-PRUNER	14.70	32.70	20.95

Table 8: Performance of SparseGPT and D-PRUNER (at 50% sparsity) on metrics of healthcare domain based on BLOOM. * denotes the model that only uses open-domain calibration data (C4) for pruning.

on MedNLI when SparseGPT only uses open-domain data, it struggles on both medical perplexity and PubMedQA. Because our method is based on Lv et al. (2023) for fine-tuning and this fine-tuning pipeline only discusses performance scores on LLaMA, Lv et al. (2023) might require a significant adaptation when we change our backbone models from LLaMA to BLOOM. It might also not work well on BLOOM-based models when we integrate the general importance as a regularization term. Therefore, we might need to switch the fine-tuning pipeline we use in order to obtain the optimal performance of D-PRUNER.

C Hyperparameters

We stick to the default values of hyperparameters for our baseline models. For D-PRUNER, in the healthcare domain, we set λ (regularization strength) and learning rate to 0.1 and 0.03. In the legal domain, we set λ and learning rate to 0.001 and 0.03.