TASP: PRESERVING TRAINING DYNAMICS IN TRANSFORMERS VIA NTK-AWARE STRUCTURED PRUNING

Mengting Ai
UIUC
Champaign, IL, USA
mai10@illinois.edu

Tianxin Wei
UIUC
Champaign, IL, USA
twei10@illinois.edu

Jingrui He
UIUC
Champaign, IL, USA
jingrui@illinois.edu

ABSTRACT

Structured pruning of large-scale Transformer models promises substantial efficiency gains by removing entire hidden units. However, such pruning often degrades accuracy more than unstructured pruning, necessitating compensation strategies such as supervised fine-tuning (SFT) or adapter modules (e.g., LoRA). In this paper, we introduce TASP (Neural Tangent Kernel-Aware Structured Pruning), a novel method that identifies and prunes low-saliency hidden units in Transformer. Our approach computes a saliency score for each weight—as the product of the weight and its partial derivative with respect to the network output—and aggregates these scores to measure the contribution of each hidden unit. We prove, via a piecewise-linear bounding argument, that pruning units with minimal saliency preserves the network's Neural Tangent Kernel (NTK) and, consequently, its training dynamics under Adam-based optimization. Empirical results on standard benchmarks confirm that TASP achieves significant model compression while maintaining training performance, offering a theoretically grounded and efficient pathway for Transformer model compression.

1 INTRODUCTION

Transformer-based (Vaswani et al., 2017) large language models (LLMs) have demonstrated remarkable performance across a range of tasks. Nevertheless, as these models grow in size and complexity, so does their computational overhead. The resulting high computational requirements make these sophisticated models inaccessible to the broader public. This limitation not only poses barriers to the democratization of AI technology but also fosters misconceptions about the techniques among the general public. A 7B parameter model in 16-bit precision occupies 14GB of memory, demanding costly GPU resources for both training and inference. This resource burden stifles accessibility, limiting broader adoption and exacerbating misconceptions about AI democratization.

Model compression, therefore, emerges as a crucial solution, aiming to make original LLMs more manageable and accessible without compromising their efficacy. Specifically, pruning (LeCun et al., 1989) focuses on eliminating non-essential weights from a model to yield a lighter, compressed variant. Current approaches fall into three categories: **(1) Unstructured pruning** methods (e.g., SparseGPT (Frantar & Alistarh, 2023), Wanda (Sun et al., 2023)[1]) eliminate individual weights but fail to achieve practical speedups due to irregular sparsity patterns incompatible with hardware accelerators;**(2) Semi-structured pruning** (e.g., 2:4 block sparsity (Zheng et al., 2024)) removes fixed weight patterns optimized for NVIDIA sparse tensor cores (Yang et al., 2023). While this improves inference efficiency, such methods cannot accelerate supervised fine-tuning (SFT), as optimizer updates disrupt the predefined sparsity structure during training;**(3) Structured pruning** techniques

[1]These two can also be applied for semi-structured pruning.

like LLM-Pruner (Ma et al., 2023) and Sheared Llama (Xia et al., 2024) remove entire neurons or layers, yet demonstrate weaker performance compared to Unstructured and Semi-structured pruning.

While unstructured pruning preserves accuracy better than structured alternatives, it offers minimal practical speedup as a trade-off(Cheng et al., 2024). Conversely, structured pruning enables hardware-friendly sparsity but requires costly compensation strategies like adapter modules (LoRA (Hu et al., 2021)) or extended retuning. Critically, all existing methods focus on post-training optimization, ignoring the SFT stage where backpropagation consumes 2x more resources than inference. Pruning *before* SFT could dramatically reduce this overhead—if done without compromising learnability.

We address this challenge through the Neural Tangent Kernel (NTK) (Jacot et al., 2018), which characterizes how parameter changes affect model outputs during training. By analyzing the NTK spectrum, we identify neurons with minimal contribution to the learning trajectory. Removing these neurons preserves the NTK's dominant modes, ensuring the pruned model retains the original's trainability during SFT. Unlike vision-focused NTK pruning (Wang et al., 2023), which assumes SGD optimization, our approach adapts to Adam's dynamics, the de facto optimizer for LLMs.

We propose **TASP** (Neural **T**angent Kernel-**A**ware **S**tructured **P**runing), the first method that synergizes structured pruning with SFT dynamics through NTK analysis. Unlike magnitude-based heuristics, TASP leverages a first-order saliency measure to quantify each neuron's contribution to the NTK spectrum, thereby identifying and eliminating redundant units that have minimal impact on model trainability. This approach enables three key advantages: (1) Hardware-efficient compression through structured sparsity, (2) Training-aware pruning that preserves critical learning dynamics, and (3) Seamless integration with standard SFT pipelines. Furthermore, our experiments on the T5 model demonstrate that TASP not only preserves the training dynamics of the original model—as verified by NTK analysis—but also maintains its performance on downstream tasks, thereby validating the effectiveness of our approach.

2 Problem Definition

2.1 Problem Formulation

We use lowercase letters to denote scalars, boldface lowercase letters to denote vectors, and boldface uppercase letters to denote matrices. The element-wise product is denoted by $\odot$. The neural network is denoted by f, parameterized by $\mathbf{W}$, and $\mathbf{x}$ represents the input data. We assume that the output $f(\mathbf{x}; \mathbf{W})$ is a single value, as is common in classification or next-token generation tasks.[2] We focus on pruning the MLP sub-layer inside a Transformer block, which is parameterized by two weight matrices: $\mathbf{W}_1 \in \mathbb{R}^{d\times h}$ and $\mathbf{W}_2 \in \mathbb{R}^{h\times d}$. For an input token $\mathbf{x} \in \mathbb{R}^d$, the MLP output is computed as

$$\mathbf{H}(\mathbf{x}) = \sigma\Big(\mathbf{x}\mathbf{W}_1\Big)\mathbf{W}_2, \tag{1}$$

where the activation function $\sigma(\cdot)$ is applied elementwise, and biases are omitted for simplicity (either because modern designs often exclude them or their contribution is marginal). While most previous work has focused on pruning entire models, we restrict our analysis to the MLP sub-layer due to its simpler structure, which facilitates clearer theoretical insights.

2.2 Neural Tangent Kernel (NTK)

SGD Perspective. Consider training via continuous-time gradient descent (GD) with learning rate η, where the parameter vector $\mathbf{W}_t$ evolves over time t. For a neural network $f(\mathbf{x}; \mathbf{W})$ with training loss $\mathcal{L}$, one can write(Lee et al., 2019a):

$$\begin{aligned}
\dot{\mathbf{W}}_t &= -\eta \nabla_{\mathbf{W}_t} f(\mathbf{x}; \mathbf{W}_t)^\top \nabla_{f(\mathbf{x};\mathbf{W}_t)} \mathcal{L}, \\
\dot{\mathcal{L}} &= \nabla_{f(\mathbf{x};\mathbf{W}_t)} \mathcal{L}^\top \nabla_{\mathbf{W}_t} f(\mathbf{x}; \mathbf{W}_t) \dot{\mathbf{W}}_t \\
&= -\eta \nabla_f \mathcal{L}^\top \Big[\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t) \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t)^\top\Big] \nabla_f \mathcal{L}.
\end{aligned}$$

[2]Without loss of generality, our analysis can be extended to the vector-output case.

The factor $\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t)\, \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t)^\top$ is called the *Neural Tangent Kernel (NTK)*(Jacot et al., 2018) under SGD, denoted

$$\widehat{\Theta}^{\mathrm{SGD}}(\mathbf{x}, \mathbf{x}) = \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}) \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W})^\top = \big\langle \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}), \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}) \big\rangle.$$

Adam (SignGD) Perspective. Modern Transformer-based language models commonly use *Adam* rather than plain SGD. Considering Adam's exact analysis is more complicated, existing work (Li et al., 2024; Zou et al., 2021; Kunstner et al., 2023; Wei et al., 2023) suggests that *Sign Gradient Descent* (SignGD) often behaves similarly in training dynamics. Hence, as a proxy for Adam, we consider a *sign-based* update:

$$\dot{\mathbf{W}}_t = -\eta \operatorname{sign}\Big(\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t)^\top \nabla_{f(\mathbf{x}; \mathbf{W}_t)} \mathcal{L} \Big).$$

By the chain rule,

$$\begin{aligned} \dot{\mathcal{L}} &= \nabla_{f(\mathbf{x}; \mathbf{W}_t)} \mathcal{L}^\top \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t) \dot{\mathbf{W}}_t \\ &= -\eta \nabla_f \mathcal{L}^\top \Big[\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t) \operatorname{sign}\big(\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}_t) \big)^\top \Big] \nabla_f \mathcal{L}. \end{aligned}$$

Following the NTK viewpoint, we define the *asymmetric SignGD kernel* as

$$\widehat{\Theta}^{\text{A-Sign}}(\mathbf{x}, \mathbf{x}) = \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}) \operatorname{sign}\big(\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}) \big)^\top = \big\langle \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}), \operatorname{sign}\big(\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}) \big) \big\rangle.$$

For simplicity, we write $\Theta = \widehat{\Theta}^{\text{A-Sign}}$ in what follows.

3 METHOD

3.1 SALIENCY SCORE AND ITS CONNECTION TO THE ADAM NTK

A saliency score is an indicator of measuring the *importance* of a weight to the model. In our method, for each weight $\mathbf{W}_{i,j}$ we define the saliency score as:

$$\mathbf{S}(\mathbf{W}_{i,j}) = \Big| \frac{\partial f(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}_{i,j}} \cdot \mathbf{W}_{i,j} \Big|. \tag{2}$$

This score reflects a *first-order* measure of how much training dynamics (as captured by the NTK) is affected when we remove that weight (i.e. set it to zero).In particular, when using Adam, the NTK is defined as: $\Theta(\mathbf{x}, \mathbf{x}) = \big\langle \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}), \operatorname{sign}\big(\nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W}) \big) \big\rangle$. For each weight $\mathbf{W}_{i,j}$, this is essentially the absolute value of the derivation $\big| \frac{\partial f}{\partial \mathbf{W}_{i,j}} \big|$. Multiplying by $\mathbf{W}_{i,j}$ further encodes how large that weight is, providing generalization to the method, avoiding the condition that even the gradient is small but the magnitude of the weights is big. A weight with a small gradient—and hence a small saliency score—contributes minimally to the NTK. Consequently, pruning such low-saliency weights should only slightly perturb Θ, leaving the essential training dynamics intact.

Proposition 3.1 (Short version of Proposition D.1)**.** *For a Transformer model, if the MLP unit is pruned with TASP, denote the NTK after pruning as $\widetilde{\Theta}$, for a sufficiently small $\epsilon > 0$, we have*

$$|\widetilde{\Theta} - \Theta| \leq O(\epsilon) \tag{3}$$

This result indicates that by selectively removing low-saliency weights, the overall NTK—and thus the training dynamics—remains nearly unchanged. This allows us to reduce model size while preserving the network's core behavior.

3.2 STRUCTURED PRUNING VIA GROUPING OF HIDDEN UNITS

To achieve practical efficiency gains, we perform structured pruning by grouping weights according to hidden units—a strategy that aligns with the dependency graph approach in (Fang et al., 2023). Specifically, we treat each hidden unit $u \in \{1, \ldots, h\}$ as a group. In the MLP sub-layer, this group consists of all weights in the u-th column of $\mathbf{W}_1$ (incoming weights) and the u-th row of $\mathbf{W}_2$

(outgoing weights). For each hidden unit, we compute a cumulative saliency score by summing the saliency scores of its associated weights:

$$S(\text{unit } u) = \sum_{i=1}^{d} \left| \frac{\partial f}{\partial (\mathbf{W}_1)_{i,u}} \cdot (\mathbf{W}_1)_{i,u} \right| + \sum_{i=1}^{d} \left| \frac{\partial f}{\partial (\mathbf{W}_2)_{u,i}} \cdot (\mathbf{W}_2)_{u,i} \right|. \tag{4}$$

We then rank all hidden units by their aggregated saliency scores and prune the bottom $v\%$ of groups. In practice, this means that any hidden unit u with a cumulative saliency $S(\text{unit } u)$ below a chosen threshold (set according to the overall sparsity ratio) is pruned—that is, its corresponding column in $\mathbf{W}_1$ and row in $\mathbf{W}_2$ are zeroed out. This grouping-based pruning ensures that the network's dependency structure is respected and that entire units are removed together, leading to real efficiency gains by reducing the MLP's intermediate dimensionality.

The remainder of our analysis and experiments will show that TASP's structured, saliency-based pruning preserves the MLP's essential capacity, including theoretical guarantees via NTK arguments, empirically performance preserved, while yielding true speedups on modern hardware.

4 Experiments

We begin by detailing the experimental setup, then compare our method to several baselines, concluding with the efficiency analysis.

4.1 Experiment Settings

We perform the experiments on t5-base (Raffel et al., 2023) on the MLP unit. All MLPs are pruned structurally by a global sparsity of 50%. We compare our method against several baselines, including foresight pruning methods originally designed for vision models—namely SNIP (Lee et al., 2019b), SynFlow (Tanaka et al., 2020), and NTK-SAP (Wang et al., 2023)—and LLM-specific pruning methods such as Wanda and LLM-Pruner. For the foresight methods, we adapt their saliency score within our framework (detailed in Appendix A.3), while for Wanda and LLM-Pruner we strictly follow the original code. In the case of LLM-Pruner, we evaluate the Element-mix variant. All experiments are conducted on a single NVIDIA V100 GPU. Additional zero-shot evaluations on LLaMa3-1B are provided in Appendix B.

4.2 Results on GLUE Dataset

Table 1 presents the fine-tuning performance of TASP and baseline methods on selected GLUE tasks. We observe the following: (1)TASP consistently outperforms the baseline methods. This empirically proves our theory that using NTK-based pruning criteria leads to better preservation of the model's learning dynamics compared to other baselines. (2)While SynFlow considers gradients with respect to the model's output, its reliance on synthetic inputs leads to suboptimal results. Similarly, NTK-SAP approximates NTK using first-order Taylor expansion in a weight-agnostic way, weakening the performance under the pre-trained language model scenario. (3)Wanda, as an unstructured method, performs worse under structured pruning constraints. Since Wanda is designed for weight-level sparsity, its performance degrades when applied to structured pruning.

Table 1: Evaluation results (accuracy) of t5-base on GLUE datasets. **Bold** indicates the best results while underline indicates the second-best.

	MRPC	CoLA	SST2	MNLI
t5-base	91.42	83.89	94.84	86.27
Magnitude	84.80	71.81	92.43	83.74
SNIP	<u>90.20</u>	<u>82.17</u>	**94.50**	85.86
SynFlow	85.29	78.04	90.94	80.97
NTK-SAP	88.24	81.30	93.35	85.16
Wanda	83.33	69.32	88.19	80.33
LLM-Pruner	<u>90.20</u>	81.20	93.58	85.37
Ours	**90.69**	**82.45**	<u>94.27</u>	**85.99**

4.3 Efficiency Analysis

Fine-tuning Efficiency. Table 2 shows the impact of pruning on computational cost. TASP reduces memory usage by 19.2% and achieves a 1.3× speedup in fine-tuning, demonstrating its practical advantage over dense models. In contrast, although SNIP utilizes a similar saliency score, it applies unstructured pruning, which is not easily exploitable by standard hardware or deep learning libraries for runtime acceleration. As a result, SNIP's fine-tuning time and memory usage remain nearly identical to the dense model.

Table 2: Model statistics after pruning, tested on one round of SST2 (batch size = 16).

	# Params (M)	Time (s)	Memory (GB)
t5-base	223.50	927.91	6.78
SNIP	166.87	928.63	6.79
TASP	166.87	732.80	5.48

5 Conclusion

In this paper, we proposed TASP, a novel method that leverages NTK analysis and saliency scores to guide structured pruning of Transformer MLP layers. TASP effectively removes redundant hidden units while preserving the training dynamics and performance of the original model, as demonstrated by our experiments on the T5 model. Our approach offers an efficient compression strategy that integrates seamlessly with standard SFT pipelines without necessitating extensive retuning.

For future work, we plan to further validate the scalability of TASP through additional experiments on the T5 family. Moreover, while our current method is applied exclusively to the simpler MLP module, we aim to extend our approach to the more complex attention modules, thereby broadening its applicability in Transformer model compression.

REFERENCES

Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 19637–19651. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/a376033f78e144f494bfc743c0be3330-Paper.pdf.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10558–10578, 2024. doi: 10.1109/TPAMI.2024.3447085.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16091–16101, June 2023.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

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

Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Finding lottery tickets in vision models via data-driven spectral foresight pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16142–16151, June 2024.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be, 2023. URL https://arxiv.org/abs/2304.13960.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24101–24116. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/987bed997ab668f91c822a09bce3ea12-Paper-Conference.pdf.

Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19 (143-155):18, 1989.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/0d1a9651497a38d8b1c3871c84528bd4-Paper.pdf.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019b. URL https://arxiv.org/abs/1810.02340.

Bingrui Li, Wei Huang, Andi Han, Zhanpeng Zhou, Taiji Suzuki, Jun Zhu, and Jianfei Chen. On the optimization and generalization of two-layer transformers with sign gradient descent, 2024. URL https://arxiv.org/abs/2410.04870.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21702–21720. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/44956951349095f74492a5471128a7e0-Paper-Conference.pdf.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

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

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6377–6389. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow, 2020. URL https://arxiv.org/abs/2002.07376.

Yite Wang, Dawei Li, and Ruoyu Sun. Ntk-sap: Improving neural network pruning by aligning training dynamics, 2023. URL https://arxiv.org/abs/2304.02840.

Tianxin Wei, Zeming Guo, Yifan Chen, and Jingrui He. NTK-approximating MLP fusion for efficient language model fine-tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 36821–36838. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/wei23b.html.

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

Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18547–18557, June 2023.

Haizhong Zheng, Xiaoyan Bai, Xueshen Liu, Z. Morley Mao, Beidi Chen, Fan Lai, and Atul Prakash. Learn to be efficient: Build structured sparsity in large language models, 2024. URL https://arxiv.org/abs/2402.06126.

Difan Zou, Yuan Cao, Yuanzhi Li, and Quanquan Gu. Understanding the generalization of adam in learning neural networks with proper regularization, 2021. URL https://arxiv.org/abs/2108.11371.

A Related Work

A.1 Notations

We use lowercase letters to denote scalars, boldface lowercase letters for vectors, and boldface uppercase letters for matrices. The element-wise product is denoted by $\odot$. Let $f(\mathbf{x}; \mathbf{W} \odot \mathbf{M})$ denote the neural network function, where $\mathbf{x}$ are the inputs, $\mathbf{W}$ the weights (or connections), and $\mathbf{M}$ the sparse mask matrix with sparsity v (density $d = 1 - v$). Additionally, let $\mathcal{L}$ be the loss function, and let

$$D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N} \subset \mathbb{R}^n \times \mathbb{R}^m$$

denote the dataset with N data points (with $\mathbf{x}$ as the input features and $\mathbf{y}$ as the labels). Finally, let $\mathcal{A}$ be the optimizer that, given the initial weights of the network before supervised fine-tuning (SFT) $\mathbf{W}^{(0)}$, returns the weights after SFT, i.e., $\mathbf{W}^{(\text{final})} = \mathcal{A}(\mathbf{W}^{(0)})$.

A.2 Post-Train Pruning

Post-train pruning compresses a fully trained dense model by removing unimportant weights or structures. A common formulation minimizes the discrepancy between the outputs of the uncompressed and pruned layers. Given an input $\mathbf{x}$, the objective is to solve:

$$\begin{aligned} &\text{argmin}_{\hat{\mathbf{W}},\mathbf{M}} \mathcal{L}\Big(\mathbf{W}^{(\text{final})}\mathbf{x},\ (\hat{\mathbf{W}} \odot \mathbf{M})\mathbf{x}\Big) \\ =&\text{argmin}_{\hat{\mathbf{W}},\mathbf{M}} \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}\Big(\mathbf{W}^{(\text{final})}\mathbf{x}_k,\ (\hat{\mathbf{W}} \odot \mathbf{M})\mathbf{x}_k\Big), \end{aligned}$$

where $\mathbf{M}$ is a mask matrix enforcing a fixed sparsity v.

Directly solving this joint optimization over $\hat{\mathbf{W}}$ and $\mathbf{M}$ is NP-hard. Consequently, popular practices include fixing the weights (i.e., setting $\hat{\mathbf{W}} = \mathbf{W}$) and searching for $\mathbf{M}$ only (one-shot pruning (Frankle & Carbin, 2019; Frantar & Alistarh, 2023; Sun et al., 2023; Chen et al., 2021)), or selecting $\mathbf{M}$ first and then optimizing $\hat{\mathbf{W}}$ (which typically requires further fine-tuning or re-training (Kwon et al., 2022; Ma et al., 2023)).

Recent works such as SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2023) have explored efficient one-shot pruning methods that eliminate weights based on local reconstruction or activation-aware criteria. In contrast, structured methods like LLM-Pruner (Ma et al., 2023) and Sheared Llama (Xia et al., 2024) remove entire rows or columns. However, these approaches generally do not consider the training dynamics explicitly and often fail to accelerate the SFT stage.

A.3 Foresight Pruning

Foresight pruning, also known as pruning before training, seeks to identify and eliminate redundant parameters at initialization, thereby reducing both training and inference costs. For a neural network f parameterized by $\mathbf{W}$ and data $(\mathbf{x}, \mathbf{y})$, the objective is formulated as:

$$\begin{aligned} &\min_{\mathbf{M}}\ \mathcal{L}\Big(f\big(\mathbf{x}; \mathcal{A}(\mathbf{W}^{(0)} \odot \mathbf{M})\big), \mathbf{y}\Big) \\ =&\min_{\mathbf{M}}\ \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}\Big(f\big(\mathbf{x}_k; \mathcal{A}(\mathbf{W}^{(0)} \odot \mathbf{M})\big), \mathbf{y}_k\Big), \end{aligned} \tag{5}$$

where $\mathcal{A}$ returns the final weights $\mathbf{W}^{(\text{final})}$. As in the post-train case, solving Equation (5) exactly is NP-hard because it involves joint optimization over the mask and the model parameters.

To bridge this gap, popular approaches define a *saliency score* $S_{i,j}$ for each weight, which estimates the impact of removing the connection $\mathbf{W}_{i,j}$. A general form of the saliency score is:

$$S(\mathbf{W}_{i,j}^{(0)}) = \frac{\partial \mathcal{I}}{\partial \mathbf{W}_{i,j}^{(0)}} \cdot \mathbf{W}_{i,j}^{(0)}, \tag{6}$$

where $\mathcal{I}$ is a function that measures the importance of the weight $\mathbf{W}$ to the network f. Once these scores are computed, the mask is obtained by selecting the top $\kappa\%$ of weights:

$$\mathbf{M}_{i,j} = \text{Top}_\kappa(S)_{i,j} = \begin{cases} 1, & \text{if } S_{i,j} \text{ is among the top } \kappa\%, \\ 0, & \text{otherwise.} \end{cases}$$

A.4 Revisiting Saliency Methods

Several representative methods use different formulations of the saliency score to approximate weight importance:

SNIP (Lee et al., 2019b) proposes a data-dependent saliency:

$$S_{\text{SNIP}} = \Big| \frac{\partial \mathcal{L}(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}_{i,j}} \cdot \mathbf{W}_{i,j} \Big|.$$

GraSP (Wang et al., 2020) employs a second-order (Hessian) metric:

$$S_{\text{GraSP}} = -\Big(\mathbf{H} \frac{\partial \mathcal{L}(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}_{i,j}} \Big) \cdot \mathbf{W}_{i,j},$$

where $\mathbf{H}$ denotes the Hessian of the loss.

SynFlow (Tanaka et al., 2020) introduces a data-agnostic approach by defining saliency on constant inputs (e.g., $\mathbf{1}$) and absolute weights:

$$S_{\text{SynFlow}} = \Big| \frac{\partial f(\mathbf{1};\, |\mathbf{W}|)}{\partial |\mathbf{W}_{i,j}|} \Big| \cdot \big| \mathbf{W}_{i,j} \big|.$$

Although SynFlow's formulation is similar to our approach, using absolute values may not fully capture the true gradient flow in the model.

NTK-SAP (Wang et al., 2023) adopts a data-agnostic perspective by injecting a small perturbation $\Delta \mathbf{W}_{i,j} \sim \mathcal{N}(0, \epsilon\, \mathbf{I})$:

$$S_{\text{NTK-SAP}} = \Big| \frac{\partial \left\| f(\mathbf{z}; \mathbf{W}) - f(\mathbf{z}; \mathbf{W} + \Delta \mathbf{W}) \right\|_2^2}{\partial \mathbf{W}_{i,j}} \Big|,$$

with $\mathbf{z}$ drawn from a standard normal distribution.

PX-Pruning (Iurada et al., 2024) introduces an auxiliary function $\mathcal{R}$ computed by two helper networks g and h (sharing the original architecture):

$$S_{\text{PX}} = \Big| \frac{\partial \mathcal{R}(\mathbf{x}, \mathbf{W}, \mathbf{a})}{\partial (\mathbf{W}_{i,j}^2)} \cdot \mathbf{W}_{i,j}^2 \Big|,$$

where

$$\mathcal{R}(\mathbf{x}, \mathbf{W}, \mathbf{a}) = g\big(\mathbf{x}^2, \mathbf{1}, \mathbf{a}\big)\, h\big(\mathbf{1}, \mathbf{W}^2, \mathbf{1}\big),$$

and $\mathbf{a}$ tracks the activation status. Backpropagation through $\mathcal{R}$ yields a saliency score for each parameter.

While these foresight methods have shown promise, they are not commonly applied to LLMs due to the models' scale and sensitivity during fine-tuning. Moreover, the inherent nature of unstructured pruning in these methods often limits their ability to reduce training costs. In contrast, our work focuses on the MLP module in LLMs and develops an NTK-aware saliency score tailored to preserve training dynamics.

B Additional Results on Zero-shot

While our primary focus is on reducing the cost of supervised fine-tuning (SFT), we also report zero-shot performance of TASP without any further tuning. Due to resource constraints, we apply a 25% sparsity ratio on Llama3-1B. The results, summarized in Table 3, indicate that although LLM-Pruner slightly outperforms TASP on the WikiText dataset, TASP achieves significantly better performance on PTB.

Table 3: Zero-shot Perplexity Results on LLama3-1B.

	WikiText	PTB
LLama3-1B	22.67	43.70
LLM-Pruner	**48.09**	91.79
TASP	48.66	**84.23**

C PSEUDOCODE OF TASP

Algorithm 1 outlines the pseudocode for our proposed TASP method. The algorithm takes as input the parameters of the MLP sub-layer (i.e., $\mathbf{W}_1$ and $\mathbf{W}_2$), the target sparsity ratio, and a mini-batch of input data.

Algorithm 1 Pruning via Group (Hidden-Unit) Saliency

Input: Model weights $\mathbf{W}_1 \in \mathbb{R}^{d\times h}$, $\mathbf{W}_2 \in \mathbb{R}^{h\times d}$, sparsity ratio r, pruning set $\mathcal{D}$, model f.
Output: Pruned weights $\widetilde{\mathbf{W}}_1 \in \mathbb{R}^{d\times h\cdot r}$, $\widetilde{\mathbf{W}}_2 \in \mathbb{R}^{h\cdot r\times d}$.
$output \leftarrow$ forward_pass$(f, \mathcal{D})$
grad $\leftarrow$ compute_gradients$(output, \mathbf{W}_1, \mathbf{W}_2)$
for $u = 1$ **to** h **do**
  $S(\text{unit } u) \leftarrow 0$
  for $i = 1$ **to** d **do**
    $S(\text{unit } u)$ += |grad$[\mathbf{W}_1(i,u)] \cdot \mathbf{W}_1(i,u)| + |\text{grad}[\mathbf{W}_2(u,i)] \cdot \mathbf{W}_2(u,i)|$
  end for
end for
$\mathcal{U}_{\text{pruned}} \leftarrow$ r%lowest_saliency_units$(S(\text{unit } u), r)$
for u in $\mathcal{U}_{\text{pruned}}$ **do**
  for $i = 1$ **to** d **do**
    $\mathbf{W}_1(i,u) \leftarrow 0$, $\mathbf{W}_2(u,i) \leftarrow 0$
  end for
end for
$\widetilde{\mathbf{W}}_1 \leftarrow \mathbf{W}_1$, $\widetilde{\mathbf{W}}_2 \leftarrow \mathbf{W}_2$
return the final weights $\widetilde{\mathbf{W}}_1, \widetilde{\mathbf{W}}_2$.

D PROOF OF PROPOSITION 3.1

Proposition D.1 (Full version of Proposition 3.1)**.** *Consider a Transformer's MLP sublayer of the form*

$$\mathbf{H}(\mathbf{X}; \mathbf{W}) = \sigma\big(\mathbf{X}\mathbf{W}_1\big)\mathbf{W}_2,$$

where $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2)$ *are the parameters (no bias),* $\mathbf{X} \in \mathbb{R}^{n\times d}$ *is a batch of inputs, and the activation function* σ *is applied elementwise. Let* Θ *denote the Neural Tangent Kernel (NTK) w.r.t.* $\mathbf{W}$*. Suppose we prune this MLP* structuredly *(removing entire hidden units) according to the saliency measure*

$$S\big(\mathbf{W}_{i,j}\big) \;=\; \Big|\, \frac{\partial f(\mathbf{X}; \mathbf{W})}{\partial \mathbf{W}_{i,j}} \;\cdot\; \mathbf{W}_{i,j} \Big|.$$

Denote the pruned parameter set as $\widetilde{\mathbf{W}}$ *(i.e. we zero entire hidden columns/rows whose group saliency is below a threshold) and let* $\widetilde{\Theta}$ *be the resulting NTK. Then for a sufficiently small* $\epsilon > 0$*, if we prune only low-saliency units so that the total pruned saliency is* ϵ*, we obtain*

$$\big|\Theta - \widetilde{\Theta}\big| \;\le\; O(\epsilon).$$

In other words, the NTK of the pruned MLP remains linearly within ϵ *of the original.*

Proof. Recall $\Theta(\mathbf{X}, \mathbf{Z}) = \langle \nabla_{\mathbf{W}} f(\mathbf{X}), \text{sign}(\nabla_{\mathbf{W}} f(\mathbf{Z})) \rangle$. Thus

$$\begin{aligned}
\left|\widetilde{\Theta} - \Theta\right| &= \left|\langle \nabla_{\widetilde{\mathbf{W}}} f(\mathbf{X}), \text{sign}(\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{Z})) \rangle - \langle \nabla_{\mathbf{W}} f(\mathbf{X}), \text{sign}(\nabla_{\mathbf{W}} f(\mathbf{Z})) \rangle\right| \\
&\leq \left|\langle \left(\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{X}) - \nabla_{\mathbf{W}} f(\mathbf{X})\right), \text{sign}\left(\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{Z})\right) \rangle\right| + \left|\langle \nabla_{\mathbf{W}} f(\mathbf{X}), \left(\text{sign}(\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{Z})) - \text{sign}(\nabla_{\mathbf{W}} f(\mathbf{Z}))\right) \rangle\right| \\
&\leq \underbrace{\|\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{X}) - \nabla_{\mathbf{W}} f(\mathbf{X})\| \cdot \|\text{sign}(\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{Z}))\|}_{\text{Term (i)}} + \underbrace{\|\nabla_{\mathbf{W}} f(\mathbf{X})\| \cdot \|\text{sign}(\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{Z})) - \text{sign}(\nabla_{\mathbf{W}} f(\mathbf{Z})\|)}_{\text{Term (ii)}}
\end{aligned}$$

Term (i). Note that $\text{sign}(\cdot)$ is a vector of ± 1 in each coordinate (ignoring coordinates exactly at zero,) hence its Euclidean norm is at most $\sqrt{2dh}$.

$$\begin{aligned}
\text{(i)} &= \|\nabla_{\widetilde{\mathbf{W}}} f(X) - \nabla_{\mathbf{W}} f(X)\| \cdot \|\text{sign}(\nabla_{\widetilde{\mathbf{W}}} f(\mathbf{Z}))\| \\
&\leq \sqrt{2dh}\, \|\nabla_{\widetilde{\mathbf{W}}} f(X) - \nabla_{\mathbf{W}} f(X)\| \\
&= \sqrt{2dh} \sum_{(i,j)\in\mathcal{P}} |\nabla_{\mathbf{W}_{i,j}} f(X)|
\end{aligned}$$

Here, since we prune the parameters according to the salience score S, we assume that for the parameters that are pruned in the set $\mathcal{P}$, we have

$$\sum_{(i,j)\in\mathcal{P}} |\nabla_{\mathbf{W}_{i,j}} f(X)| \cdot |\mathbf{W}_{i,j}| \leq \epsilon$$

Assuming weights are bounded $|\mathbf{W}_{i,j}| \geq c$ for pruned parameters since the model is pre-trained, we then have

$$\sum_{(i,j)\in\mathcal{P}} |\nabla_{\mathbf{W}_{i,j}} f(X)| \leq \frac{\epsilon}{c}$$

Hence

$$\text{(i)} \leq \frac{\sqrt{2dh}}{c}\epsilon$$

Term (ii). Similarly, $\left\|\text{sign}(\nabla_{\widetilde{\mathbf{W}}} f(Z)) - \text{sign}(\nabla_{\mathbf{W}} f(Z))\right\| = \|\text{sign}(\nabla_{\mathbf{W}_{\mathcal{P}}} f(Z))\| \leq \sqrt{2dh}$. As for $\|\nabla_{\mathbf{W}} f(X)\|$, since the activation function σ (usually ReLU) is 1-Lipschitz, and $\mathbf{W}$ is under-controlled with weight-decay in Adam, it is safe to assume $\|\nabla_{\mathbf{W}} f(X)\| \leq G$ for a small constant G. Consequently,

$$\text{(ii)} \leq G\sqrt{2dh}$$

Combine Term (i) & (ii).

$$\begin{aligned}
\left|\widetilde{\Theta} - \Theta\right| &\leq \text{Term (i)} + \text{Term (ii)} \\
&\leq \sqrt{2dh}(\frac{\epsilon}{c} + G) \\
&= O(\epsilon)
\end{aligned}$$

$\diamondsuit$