JD-LLM: Jacobian Deviation for Efficient Layer Pruning in Large Language Models

Anonymous ACL submission

Abstract

Structured pruning is critical for reducing the inference cost of large language models, yet most criteria rely on heuristics that only indirectly capture a layer's function. We propose *Jacobian Deviation from Identity* (JD), a derivative-based criterion that measures how much a Transformer layer alters its input, estimated efficiently via Jacobian–vector products (JVP) with finite-difference or forward-mode approximations. Experiments on $\text{RoBERTa}_{\text{base}}$ (GLUE) and T5-Base (e-SNLI, CQA, ANLI, SVAMP) reveal stable sensitivity profiles—early–mid peaks with late decay in RoBERTa, and strong concentration in the first layers of T5. Guided by these profiles, pruning four to six layers preserves or even improves accuracy while outperforming knowledge-entropy pruning, random LayerDrop, energy distance pruning, and SlimLLM-style methods. Our results show that near-identity Jacobians signal empirical redundancy, making JD a simple, scalable, and principled criterion for architecture-aware pruning.

1 Introduction

Large language models (LLMs) achieve strong performance across a wide range of Natural Language Processing (NLP) tasks but come with substantial inference costs. To make deployment more practical, recent research has focused on *structured pruning* of Transformer components—such as attention heads, channels, blocks, or entire layers—in order to achieve speedups with minimal accuracy degradation (Fan et al., 2020a; Lu et al., 2024a; Ling et al., 2024; Sun et al., 2023; Guo et al., 2025; Chen et al., 2025; Gromov et al., 2025). Most existing pruning criteria, however, are *activation- or parameter-driven*: they rely on correlations between hidden states, such as Singular Vector Canonical Correlation Analysis (SVCCA), Centered Kernel Alignment (CKA), or cosine similarity (Raghu et al., 2017; Kornblith et al., 2019; Chen et al., 2025; Guo et al., 2025), as well as magnitude heuristics or Fisher/Taylor approximations defined in parameter space. These proxies often correlate only loosely with a layer's *functional role* in the residual architecture.

We argue that the right question is: *how much does a layer actually transform its input?* Motivated by theory on Jacobian geometry and dynamical isometry where near-identity mappings propagate information with little distortion (Pennington et al., 2017; Schoenholz et al., 2017; Yang et al., 2021), we introduce Jacobian Deviation from Identity (JD) as a *derivative-based, representation-space* criterion for layer importance. For a layer map f_l, the Jacobian $J_l = \partial h_{l+1}/\partial h_l$ quantifies local sensitivity; when $J_l \approx I$, the layer behaves like a skip connection and is a natural pruning target. We estimate JD efficiently via Jacobian–vector products (JVPs) using two practical estimators: finite differences (FD-JVP) and forward-mode automatic differentiation (AD-JVP). This yields a scalable, data-aware score that directly captures first-order functional change, complementing saliency and similarity measures.

We validate JD on two representative architectures and diverse tasks. For the encoder-only $\text{RoBERTa}_{\text{base}}$ on GLUE, FD-JVP and AD-JVP reveal a consistent *early–mid peak and late decay* profile, indicating that the deepest 2–3 layers are most redundant. Pruning four to six layers by JD preserves accuracy and often matches or surpasses the unpruned baseline, outperforming knowledge-entropy pruning, random LayerDrop, and SlimLLM-style structured pruning. For the encoder–decoder T5-Base on e-SNLI, CQA, ANLI, and SVAMP, sensitivity is sharply concentrated in the first 1–2 layers, with later decoder blocks exhibiting near-zero JD; JD-guided pruning of four decoder layers again maintains or improves accuracy, with AD-JVP providing the most stable gains. These findings highlight that redundancy

profiles are *model-specific* (encoder-only vs. encoder–decoder) and that JD offers a principled way to tailor pruning strategies accordingly.

The Contributions of this paper are as follows:

- We propose JD as a functionally grounded, architecture-agnostic criterion for structured *layer* pruning in Transformers.
- We develop two scalable JD estimators (FD-JVP and AD-JVP) that avoid explicit Jacobian formation while remaining data-aware through minibatch probing.
- We present a systematic analysis of JD profiles for $\text{RoBERTa}_{\text{base}}$ (GLUE) and T5-Base (e-SNLI, CQA, ANLI, SVAMP), revealing distinct redundancy patterns across architectures.
- We demonstrate that JD-based pruning retains or improves task performance while removing four to six layers, outperforming knowledge entropy-based pruning, random LayerDrop, energy distance pruning, and SlimLLM-style structured pruning.

Together, these results bridge theory and practice: layers with Jacobians near identity are empirically redundant, and JD provides a simple, reliable signal for compressing LLMs without sacrificing core representational capacity.

2 Related Work

Structured pruning in LNNs. Structured pruning has progressed from unstructured weight removal (LeCun et al., 1989; Hassibi and Stork, 1992; Han et al., 2016) to channel/filter pruning (Li et al., 2017; Molchanov et al., 2017a; He et al., 2017), block- and layer-level pruning (Fan et al., 2020a), and Transformer-specific head, MLP, and layer pruning (Michel et al., 2019; Voita et al., 2019; Lagunas et al., 2021; Frantar and Alistarh, 2023). Recent studies show that many LLM layers are redundant and can be removed with limited degradation (Men et al., 2024; Chen et al., 2024; Yang et al., 2024; Lu et al., 2024b; Siddiqui et al., 2024; Sun et al., 2023; Guo et al., 2025; Gromov et al., 2025). Our method differs by introducing a derivative-based functional criterion: Jacobian deviation from identity.

Representation similarity and Jacobian-based criteria. Many methods quantify layer redundancy through activation similarity or stability, including SVCCA, CKA, hidden-state cosine similarity, and perturbation-based measures (Raghu et al., 2017; Kornblith et al., 2019; Chen et al., 2025; Morcos et al., 2018). These are correlation-based criteria. By contrast, Jacobian deviation from identity (JDI) measures local functional sensitivity through the layer Jacobian $J_l = \partial h_{l+1}/\partial h_l$, directly testing whether a layer behaves like a near pass-through ($J_l \approx I$) or a substantial transformation.

Jacobian analysis and pruning signals. Jacobian properties play a central role in deep learning theory. Dynamical isometry links Jacobians with stable optimization and information propagation (Pennington et al., 2017; Schoenholz et al., 2017; Yang et al., 2021), while Jacobian norms are also tied to robustness and generalization (Cisse et al., 2017; Miyato et al., 2018; Sokolić et al., 2017; Rifai et al., 2011; Hoffman et al., 2019). Related pruning criteria based on gradients, Taylor expansions, and Fisher information estimate loss sensitivity in parameter space (Molchanov et al., 2017b; Lee et al., 2019). JDI instead evaluates functional sensitivity in representation space. Recent work on Jacobian spectra, scalable JVP computation, and Jacobian-based sensitivity further supports this perspective (Zhang and Stadie, 2020; Balestriero and Baraniuk, 2021; Dadoun et al., 2025).

Compression and positioning. Modern LLM/VLM compression combines quantization, sparse or structured pruning, and layer dropping/replacement (Frantar et al., 2023; Dettmers et al., 2023; Frantar and Alistarh, 2023; Ma et al., 2023; Chen et al., 2025; Fan et al., 2020a). Most existing methods rely on parameter- or activation-based proxies. To our knowledge, no prior method explicitly ranks Transformer layers by Jacobian deviation from identity and prunes the lowest-scoring layers. JDI bridges Jacobian theory and practical layer compression by providing a first-order, architecture-agnostic criterion aligned with residual Transformer design.

3 Proposed Method

3.1 Jacobian Deviation as a Pruning Criterion

Layer pruning reduces inference cost by eliminating layers that contribute little to task perfor-

mance. Conventional pruning criteria—such as magnitude- or entropy-based metrics—focus on parameter statistics or activation sparsity, but these do not directly capture the *functional transformation* a layer performs. Prior work suggests that Jacobian structure is closely tied to model complexity and generalization (Novak et al., 2018; Papyan, 2019), while representational similarity has been applied to detect redundancy in Transformers (Michel et al., 2019; Fan et al., 2020b). The theoretical justification for using JD as a pruning signal is provided in Appendix A. Motivated by these observations, we introduce JD as a functionally grounded pruning criterion that directly measures how much a layer deviates from an identity mapping, thereby indicating whether it performs meaningful transformation or acts as a redundant pass-through.

Formally, let layer l map $h_l \mapsto h_{l+1} = f_l(h_l)$ with Jacobian

$$J_l = \frac{\partial f_l(h_l)}{\partial h_l}. \tag{1}$$

If $J_l \approx I$, the layer acts as a near-identity mapping and is likely redundant. Significant deviation from identity implies meaningful representational change and justifies retention. We therefore define the importance of a layer as

$$\mathrm{JD}(l) = \mathbb{E}_{x\sim\mathcal{B}}\big[\, \|J_l(x) - I\|_F^2 \,\big], \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\mathcal{B}$ is a probe batch.

3.2 Approximating Jacobian Deviation

Directly forming J_l is infeasible for large models. Instead, we approximate $\|J_l - I\|_F^2$ using Hutchinson-style estimators (Hutchinson, 1989):

$$\|J_l - I\|_F^2 = \mathbb{E}_v\big[\|(J_l - I)v\|_2^2\big], \tag{3}$$

where $v \sim \mathcal{N}(0, I)$. Thus, JD estimation reduces to computing Jacobian–vector products (JVPs). We employ two strategies, Finite differences (FD) and Forward-mode Automatic Differentiation (AD).

3.2.1 FD JVP Approximation

FD is a straightforward approach to approximate JVPs. For a probe vector $v \sim \mathcal{N}(0, I)$, we estimate (Stewart et al., 2021)

$$J_l v \;\approx\; \frac{f_l(x + \epsilon v) - f_l(x)}{\epsilon}, \tag{4}$$

where ϵ is a small step size. The deviation is then computed as

$$(J_l - I)v \;\approx\; \frac{f_l(x + \epsilon v) - f_l(x)}{\epsilon} - v. \tag{5}$$

Averaging over multiple random probes yields an unbiased Monte Carlo estimate of $\|J_l - I\|_F^2$ (Hutchinson, 1989; Pearlmutter, 1994). This method is simple and memory-efficient, but may suffer from numerical instability when ϵ is too small or the layer output scale is large.

3.2.2 AD JVP Approximation

AD efficiently computes JVPs without forming the Jacobian. For a layer $f_l : \mathbb{R}^d \to \mathbb{R}^d$ with input x and probe vector v, the JVP is (Stewart et al., 2021)

$$J_l v \;=\; \frac{d}{d\epsilon} f_l(x + \epsilon v)\Big|_{\epsilon=0}, \tag{6}$$

i.e., the directional derivative of f_l at x in direction v. Unlike finite-difference methods, AD evaluates this exactly (up to floating-point error) by propagating *primal–tangent* pairs through the computation graph, e.g.

$$(Wx+b,\, Wv), \quad (\tanh(x),\, (1-\tanh^2(x))\odot v).$$

where $\odot$ denotes elementwise (Hadamard) multiplication.

For pruning, we estimate

$$(J_l - I)v = J_l v - v, \tag{7}$$

and average $\|(J_l - I)v\|^2$ across random probe vectors. Each JVP requires only one forward-mode sweep, reducing cost from $O(d^2)$ to $O(d)$ per probe—tractable even for Transformers with $d > 10^4$.

AD JVPs are step-size free, and scale linearly with d, making them well-suited for analyzing layer sensitivity and pruning in large models (Pearlmutter, 1994; Griewank and Walther, 2008; Baydin et al., 2018).

Discussion. Both FD-JVP and AD-JVP provide unbiased estimators of Jacobian deviation under random probes. FD-JVP is lightweight and broadly applicable, but requires careful step-size selection, whereas AD-JVP achieves higher numerical precision by exploiting exact forward-mode derivatives.

3.3 JVP Approximations in MHSA

Let $f_l : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ be the Multi-Head Self-Attention (MHSA) map at layer l. Given probe $V \in \mathbb{R}^{n\times d}$, the JVP is $J_l V := \frac{\partial f_l(X)}{\partial X}[V]$.

FD-JVP (black-box). With step $\varepsilon > 0$,

$$J_l V \approx \frac{f_l(X+\varepsilon V) - f_l(X)}{\varepsilon} \quad \text{(forward)}, \tag{8}$$

$$J_l V \approx \frac{f_l(X+\varepsilon V) - f_l(X-\varepsilon V)}{2\varepsilon} \quad \text{(central)}. \tag{9}$$

Forward uses one extra pass; central uses two and reduces truncation from $O(\varepsilon)$ to $O(\varepsilon^2)$. A stable choice in fp32 is $\varepsilon = \eta \, \|X\|_F / (\|V\|_F + \delta)$ with $\eta \in [10^{-6}, 10^{-4}]$, $\delta = 10^{-12}$. For consistency: disable dropout, reuse masks, fix LayerNorm stats, and keep RNG fixed across evaluations.

AD-JVP (forward mode). For MHSA with H heads and per-head dim d_h (Vaswani et al., 2017),

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \tag{10}$$

$$A = \frac{1}{\sqrt{d_h}} QK^\top, \quad P = \mathrm{softmax}(A), \tag{11}$$

$$Z = PV, \;\; Y = \mathrm{Concat}(Z_1, \ldots, Z_H) W_O \tag{12}$$

Propagate primal–tangent pairs $(u, \dot{u})$:

$$\dot{Q} = \dot{X} W_Q, \;\; \dot{K} = \dot{X} W_K, \;\; \dot{V} = \dot{X} W_V, \tag{13}$$

$$\dot{A} = \frac{1}{\sqrt{d_h}} (\dot{Q} K^\top + Q \dot{K}^\top), \tag{14}$$

$$\dot{P} = \big(\mathrm{Diag}(P) - PP^\top\big) \dot{A} \;\; \text{(row-wise)}, \tag{15}$$

$$\dot{Z} = \dot{P} V + P \dot{V}, \tag{16}$$

$$\dot{Y} = \mathrm{Concat}(\dot{Z}_1, \ldots, \dot{Z}_H) W_O. \tag{17}$$

The output tangent $\dot{Y}$ equals $J_l V$. Cost is $\sim$ one forward sweep per probe; no step-size tuning.

Cost/accuracy and when to use what. FD-JVP is implementation-agnostic but step-size–sensitive; central differences improve bias at $2\times$ cost. AD-JVP is exact (up to fp error) and typically matches one forward pass per probe. When forward-mode AD is available, use AD-JVP; otherwise, FD-JVP with the relative step above works well.

3.4 Interpretation and Advantages

Layers with *low JD* behave almost like skip connections and can be pruned with minimal effect on accuracy, while *high JD* values identify layers that perform essential transformations. This view is consistent with empirical findings that certain middle layers in Transformers collapse to near-identity mappings (Fan et al., 2020b; Frantar and Alistarh, 2023). Compared to conventional heuristics, JD offers three advantages: (i) it is *functionally grounded*, directly quantifying representational change; (ii) it is *data-aware*, adapting to the task distribution via minibatch probes; and (iii) it operates at the *layer level*, complementing structured pruning approaches for LLMs.

Jacobian Deviation pruning provides a principled, information-theoretic measure of layer importance. By leveraging efficient JVP approximations, it scales to modern Transformers and identifies layers whose contribution is negligible, enabling compression without loss of core representational capacity.

4 Experimental Results

4.1 Layer Pruning of RoBERTa

We evaluate Jacobian Deviation (JD) during full fine-tuning of RoBERTa (Liu et al., 2019) on the GLUE benchmark (Wang et al., 2018). Our experiments use $\text{RoBERTa}_{\text{base}}$, which shares the same architecture as $\text{BERT}_{\text{base}}$: 12 transformer encoder layers with hidden size 768, 12 attention heads, and a total of 125 million parameters. $\text{RoBERTa}_{\text{base}}$ is a widely adopted backbone for natural language understanding and serves as a standard baseline on GLUE.

Figures 1 and 2 plot FD-JVP and AD-JVP values across layers during six epochs of full fine-tuning on the eight GLUE tasks. Across tasks, both estimators exhibit a consistent trend: sensitivity rises from the embedding layer to an early–mid-depth peak ($\approx$ layers 2–4), then steadily decays toward deeper layers. While the epoch-to-epoch curves largely shift in parallel, the overall profile remains stable, indicating that fine-tuning primarily rescales sensitivity rather than altering the order of important layers. AD-JVP provides smoother, more stable traces than FD-JVP, which displays dataset-specific fluctuations and occasional late-layer outliers. A notable exception is STS-B, where AD-JVP shows a sharp increase in the final block, reflecting stronger supervision from the regression objective.

These observations suggest that the JVP profile is largely model-determined and dataset-invariant. Fine-tuning affects magnitudes but does not change which layers are most influential. This makes the deepest 2–3 layers natural candidates for pruning or freezing in classification tasks, while early and mid-depth layers should generally be preserved. For regression tasks such as STS-B, retaining the final block is essential. Between the two estimators, AD-JVP is preferable in practice: it matches the global shape of FD-JVP while avoiding its numerical instabilities, making it a more reliable redundancy

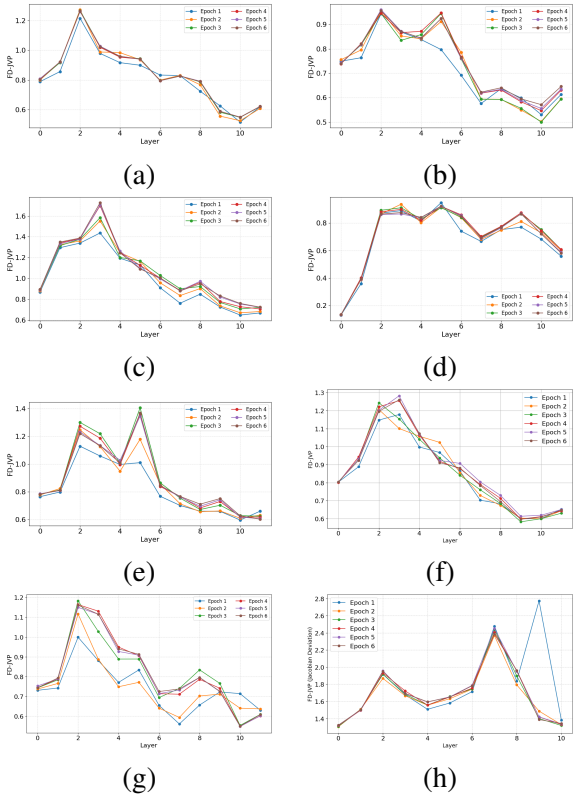


Figure 1: The FD-JVP versus layer index in RoBERTa$_{\text{base}}$ with full fine tuning for 6 epochs using eight datasets on the GLUE benchmark, (a) MNLI, (b) MRPC, (c) SST-2, (d) CoLA, (e) QNLI, (f) QQP, (g) RTE, (h) STS-B.

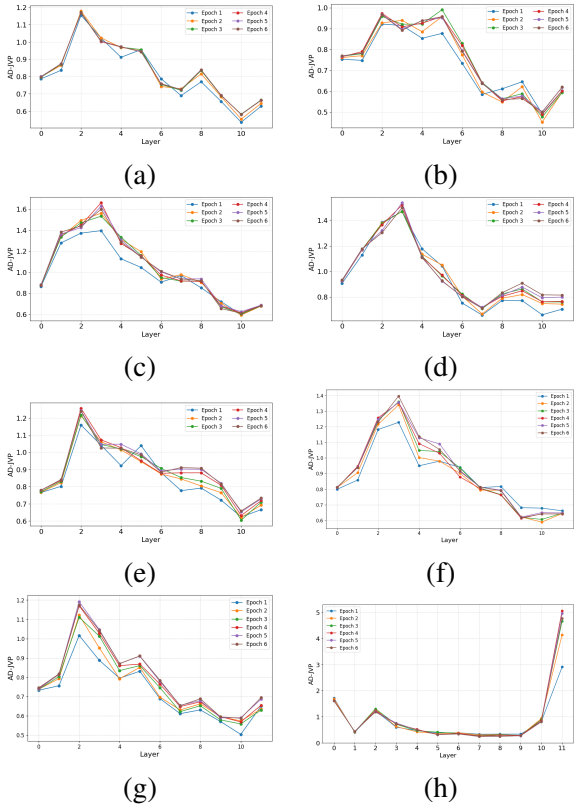


Figure 2: The AD-JVP versus layer index in RoBERTa$_{\text{base}}$ with full fine tuning for 6 epochs using eight datasets on the GLUE benchmark, (a) MNLI, (b) MRPC, (c) SST-2, (d) CoLA, (e) QNLI, (f) QQP, (g) RTE, (h) STS-B.

score.

In Gromov et al. (2025), it showed that pruning more than 50% of layers in LLMs leads to severe performance degradation, so we prune 4-6 of the 12 transformer layers after 6 fine-tuning epochs according to each layer's JVP. Table 1 reports post-pruning GLUE results, comparing five criteria: (1) pruning by lowest JD from FD-JVP or AD-JVP, (2) pruning by highest knowledge entropy (KE) (Kim et al., 2025), (3) pruning by energy distance (ED) between input and output of layers (Soltani et al., 2022), (4) random structured LayerDrop (Fan et al., 2020a), and (5) SlimLLM structured pruning (Guo et al., 2025). After pruning, models are fine-tuned for 5 additional epochs. Results show that JD and AD-based pruning methods maintain competitive or superior performance relative to other structured pruning baselines across multiple GLUE tasks.

The results in Table 1 highlight the effectiveness of Jacobian-based pruning relative to alternative structured pruning methods. KE pruning performs poorly across all tasks, with severe degradation in accuracy and correlation scores, suggesting that KE is not a reliable redundancy measure. Random structured LayerDrop retains moderate performance but consistently trails behind the no-pruning baseline. SlimLLM pruning achieves more competitive results, yet still lags in several tasks. In contrast, both FD-JVP and AD-JVP pruning preserve task performance remarkably well: FD pruning even surpasses the fully fine-tuned baseline on MNLI, MRPC, and SST-2, while AD pruning achieves the highest scores on MNLI, QQP, and RTE. The overall performance drop is minimal compared to the baseline, demonstrating that Jacobian deviation is a principled signal for identifying prunable layers.

4.2 Layer Pruning of T5-Base

In addition to RoBERTa$_{\text{base}}$, we study the Text-to-Text Transfer Transformer (T5) (Raffel et al., 2019), which frames all NLP tasks as text-to-text. Among its variants, we focus on T5-Base (220M parameters) for our investigation of *redundancy*

Table 1: GLUE benchmark accuracy (or correlation for STS-B) for RoBERTa$_{\text{base}}$ under different pruning depths. The *No Pruning* case is the baseline without removing any layers.

Pruning	Method	MNLI	MRPC	SST-2	CoLA	QNLI	QQP	RTE	STS-B
0 Layers	No Pruning	0.778	0.860	0.920	**0.828**	**0.874**	0.837	0.751	**0.905**
4 Layers	KE Pruning	0.367	0.684	0.729	0.691	0.505	0.632	0.527	0.313
	ED Pruning	0.784	0.696	0.873	0.795	0.844	0.813	0.672	0.888
	Layer Dropout	0.691	0.824	0.919	0.691	0.837	0.800	0.726	0.894
	SlimLLM	0.766	0.809	0.898	0.809	0.855	0.836	0.679	0.904
	Our FD Pruning	0.793	**0.870**	**0.927**	0.801	0.860	0.838	0.747	0.890
	Our AD Pruning	**0.803**	0.858	0.911	0.817	0.864	**0.839**	**0.762**	0.902
5 Layers	SlimLLM	0.739	0.809	0.898	0.808	0.874	0.829	0.726	0.893
	Our FD Pruning	0.777	0.860	0.927	0.794	0.869	0.835	0.729	0.878
	Our AD Pruning	0.782	0.843	0.913	0.820	0.863	0.836	0.747	0.899
6 Layers	SlimLLM	0.727	0.762	0.891	0.797	0.865	0.827	0.697	0.893
	Our FD Pruning	0.777	0.819	0.913	0.794	0.868	0.834	0.718	0.881
	Our AD Pruning	0.776	0.828	0.920	0.818	0.856	0.832	0.744	0.885

scores. T5-Base is an encoder–decoder model with 12 encoder and 12 decoder layers: each encoder block combines multi-head self-attention and a feed-forward network with residuals and layer normalization, while each decoder block adds cross-attention to the encoder outputs alongside masked self-attention and feed-forward sublayers. All attention modules use 12 heads with model dimension 768, enabling expressive sequence transduction and conditioning.

We evaluate using four established benchmarks—e-SNLI (Camburu et al., 2018), ANLI (Nie et al., 2019), CommonsenseQA (CQA) (Talmor et al., 2019), and SVAMP (Patel et al., 2021)—chosen to span language understanding and reasoning scenarios. All samples are preprocessed with the `T5` tokenizer for consistent training/evaluation. Following (Hsieh et al., 2023), we refer to e-SNLI, CQA, ANLI, and SVAMP as CoT (Chain-of-Thought) benchmarks.

Figures 3 and 4 show the FD-JVP and AD-JVP profiles across layers of T5-base during six epochs of full fine-tuning on the GLUE benchmark. Unlike RoBERTa, T5 exhibits highly skewed sensitivity patterns: both FD-JVP and AD-JVP peak sharply at the input embedding and first encoder block, followed by a rapid decay toward near-zero values in subsequent layers. This indicates that most representational transformations are concentrated in the earliest layers, with later layers contributing comparatively little in terms of Jacobian deviation. While AD-JVP traces appear smoother and more numerically stable than FD-JVP, both estimators agree on the dominant role of the first two layers. The stability of this trend across datasets and epochs further confirms that the profile is model-driven rather than task-specific. These results suggest that, for pruning or compression, the deeper layers of T5-base are natural candidates for removal, as they exhibit minimal sensitivity, while the early layers should be preserved to retain performance.

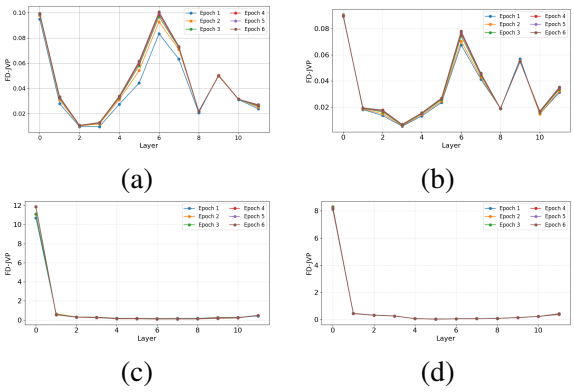


Figure 3: The FD-JVP versus layer index in T5-base with full fine tuning for 6 epochs using four datasets, (a) e-SNLI, (b) CQA, (c) ANLI, (d) SVAMP.

We next prune 4-6 of the twelve decoder layers in T5-Base using six strategies: FD and AD-based pruning, knowledge entropy pruning (KE-Pruning) (Kim et al., 2025), random Layer-

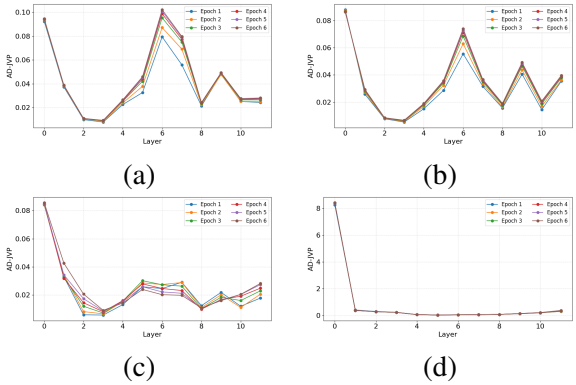


Figure 4: The AD-JVP versus layer index in T5-base with full fine tuning for 6 epochs using four datasets, (a) e-SNLI, (b) CQA, (c) ANLI, (d) SVAMP.

Table 2: T5-Base accuracy on four benchmarks with pruning 4–6 layers in the T5 decoder. The *No Pruning* case is the baseline without removing any layers.

Pruning	Approach	e-SNLI	CQA	ANLI	SVAMP
0 Layer	No Pruning	0.536	0.577	0.405	0.340
4 Layers	KE-Pruning	0.531	0.541	0.318	0.320
	Layer Drop	0.525	0.474	0.290	0.305
	SlimLLM	0.532	0.567	0.369	0.270
	FD Pruning	0.537	0.577	0.417	0.365
	AD Pruning	0.542	0.579	**0.439**	0.369
5 Layers	SlimLLM	0.531	0.563	0.386	0.325
	FD Pruning	0.533	0.577	0.427	0.390
	AD Pruning	0.537	0.577	0.4150	**0.415**
6 Layers	SlimLLM	0.530	0.571	0.387	0.255
	FD Pruning	**0.544**	0.578	0.423	0.385
	AD Pruning	0.529	**0.582**	0.408	0.380

Drop (Fan et al., 2020a), SlimLLM (Guo et al., 2025), and a no-pruning baseline for comparison. Table 2 reports accuracy on four benchmarks: e-SNLI, CQA, ANLI, and SVAMP.

Table 2 shows that JD-based pruning (FD and AD) consistently outperforms other structured pruning approaches across the four benchmarks. KE-Pruning and random LayerDrop lead to significant performance drops, with accuracy degrading especially on CQA and ANLI, indicating that these methods are unreliable indicators of redundancy in T5. SlimLLM maintains moderate accuracy, competitive with the baseline on CQA but weaker on other datasets. In contrast, FD-Pruning matches or exceeds the no-pruning baseline on e-SNLI and achieves notable gains on ANLI and SVAMP. AD-Pruning further improves stability, achieving the highest overall scores, including clear improvements on e-SNLI and ANLI.

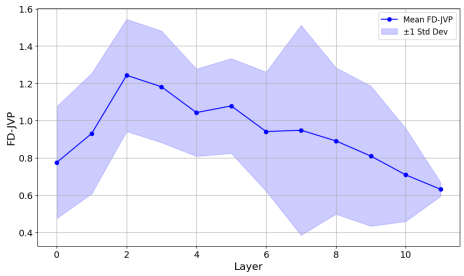
These results confirm that redundancy score–based pruning guided by Jacobian deviation is an effective and principled strategy for compressing T5-Base while preserving or even enhancing performance. In particular, AD-Pruning proves to be the most robust, offering consistent accuracy improvements across diverse tasks. This reinforces the conclusion that Jacobian sensitivity is a more reliable pruning signal than entropy-based or stochastic approaches for encoder-decoder architectures such as T5.

Interestingly, the pruning dynamics in T5-base differ markedly from those observed in RoBERTa-base. In RoBERTa, Jacobian sensitivity peaks at early–mid layers and decays gradually, suggesting that the deepest layers are the most redundant. By contrast, T5-base concentrates most of its representational transformations in the first one or two layers, with subsequent decoder layers contributing far less. As a result, pruning in T5 can safely target later decoder blocks without sacrificing performance, whereas in RoBERTa careful preservation of early–mid layers is critical. This contrast highlights how encoder-only and encoder–decoder architectures exhibit distinct redundancy profiles, underscoring the importance of model-specific sensitivity analysis when designing pruning strategies.

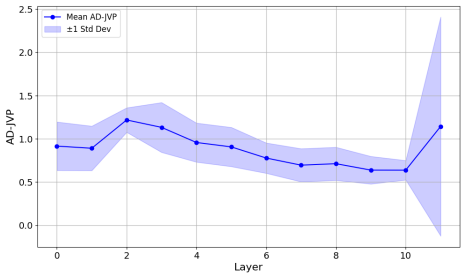
4.3 Model and Dataset Effects on Jacobian Deviation

In Figs. 5, we report the mean and standard deviation (std) of JVP across all RoBERTa-base experiments on GLUE (totally 48 experimental collections). The two panels show the layer-wise profiles of FD-JVP and AD-JVP. In both cases, the mean curve reflects an intrinsic property of the architecture: early-to-middle layers exhibit larger Jacobian-deviation magnitudes, whereas deeper layers progressively stabilize with reduced sensitivity. This pattern indicates that the mean is largely model-determined, capturing how RoBERTa-base consistently transforms representations across layers irrespective of the dataset. In contrast, the shaded std bands are appreciably wider—especially from middle to late layers—showing that cross-dataset differences account for most of the variability. Put differently, the mean traces the canonical transformation dynamics of RoBERTa-base, while the spread is dataset-driven, revealing how task signals from different GLUE datasets perturb the layerwise Jacobian sensitivity.

In Figs. 6, we plot the mean and std of JVP for

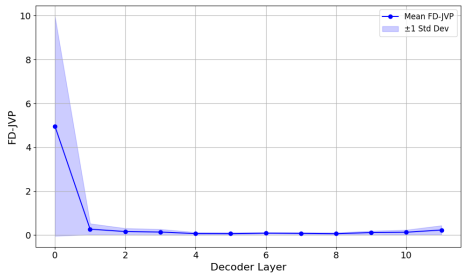


(a)

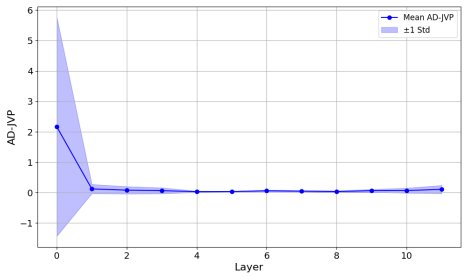


(b)

Figure 5: Mean and standard deviation of JVP versus layer index for RoBERTa-base with full fine-tuning over six epochs on the GLUE benchmark (eight datasets; 48 experimental collections). (a) FD-JVP. (b) AD-JVP.



(a)



(b)

Figure 6: Mean and standard deviation of JVP versus layer index for T5-base with full fine-tuning over six epochs on four chain-of-thought datasets (24 experimental collections). (a) FD-JVP. (b) AD-JVP.

T5-base, aggregated over 24 experimental collections from four chain-of-thought (CoT) datasets. Both FD-JVP and AD-JVP display a steep drop after the embedding layer (Layer 0), with subsequent layers remaining close to zero. This indicates that most representational distortion is concentrated at the input stage, while deeper layers of T5 stabilize and propagate information more consistently. For FD-JVP, the mean peaks at the input and quickly converges to small positive values downstream, suggesting that forward-difference perturbations are largely absorbed early. AD-JVP follows the same trend with slightly noisier tails. In both panels, the largest variance appears at Layer 0, highlighting that dataset-specific effects dominate the embedding interface, whereas middle and deeper layers remain relatively consistent across datasets.

5 Conclusions and Future Work

We introduced JD as a derivative-based, representation-space criterion for structured layer pruning in Transformers. Using Jacobian–vector products via FD-JVP and AD-JVP, JD avoids explicit Jacobian construction while remaining data-aware through minibatch probes. Across $\text{RoBERTa}_{\text{base}}$ on GLUE and T5-Base on e-SNLI, CQA, ANLI, and SVAMP, JD reveals stable model-specific sensitivity profiles that lead to effective pruning policies: an early–mid peak with late decay for RoBERTa and strong concentration in the first layers for T5. Guided by these profiles, pruning four to six layers preserves, and in some cases improves, accuracy relative to no-pruning baselines, outperforming KD pruning, ED pruning, random LayerDrop, and SlimLLM-style structured pruning. AD-JVP is generally the more stable estimator, while FD-JVP provides a lighter alternative that tracks similar trends. JD remains a first-order local criterion and may miss higher-order or long-range cross-layer interactions; it also depends on the probe distribution and can be influenced by normalization and residual scaling.

A natural next step is to use JD during training, for example to regularize layers toward near-identity mappings or to dynamically gate or freeze blocks during fine-tuning. JD may also support budgeted test-time pruning, early-exit strategies, and parameter-efficient adaptation by identifying where adapters or LoRA modules should be placed. Extending JD to finer granularity, such as MHSA, FFN, cross-attention, or channel-level pruning, could enable more aggressive compression. Finally, combining JD with quantization, distillation, or compiler-level optimization are promising directions for future work.

6 Limitations

JD provides a local, first-order measure of layer sensitivity around the probed activations, so it may miss nonlocal or higher-order effects such as curvature, long-range cross-layer interactions, or layers that matter only for rare inputs. Because JD is data-aware, its rankings depend on the probe distribution and the stage of fine-tuning at which scores are measured; under-represented domains or objectives, such as regression, may therefore shift the observed sensitivity patterns. In addition, residual connections and normalization can rescale activations and derivatives, meaning that near-identity Jacobians may sometimes reflect parameterization or scaling effects rather than true semantic irrelevance, while large JD values may partly arise from amplification. Finally, the estimator introduces practical tradeoffs: FD-JVP is sensitive to step size and can produce outliers, whereas AD-JVP is generally more stable but depends on forward-mode JVP support and careful implementation, especially under mixed precision.

7 Ethical Considerations

JD-based pruning aims to reduce inference cost, which can lower energy consumption and improve accessibility, but it also introduces several ethical concerns. Because pruning is guided by probe data, under-representation of certain groups, dialects, or domains may disproportionately degrade performance for those populations, making subgroup and slice-based evaluations important before deployment. In addition, compression can affect robustness, calibration, and behavior under distribution shift even when average accuracy remains stable, so pruned models should be carefully tested and should not replace existing safeguards in high-stakes applications. JD estimation also relies on probe batches, which may contain sensitive data, requiring appropriate data governance, privacy protection, and security practices throughout the pruning pipeline. Finally, while making models cheaper to run can broaden beneficial access, it may also lower the barrier to misuse at scale, underscoring the need for responsible access control and monitoring.

References

Randall Balestriero and Richard G. Baraniuk. 2021. Fast jacobian-vector product for deep networks. *CoRR*, abs/2104.00219.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2018. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, volume 31.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2024. Streamlining redundant layers to compress large language models. *arXiv preprint arXiv:2403.19135*.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025. Streamlining redundant layers to compress large language models. *ICLR*.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*.

Benjamin Dadoun, Soufiane Hayou, Hanan Salam, Mohamed El Amine Seddik, and Pierre Youssef. 2025. On the stability of the jacobian matrix in deep neural networks. *Preprint*, arXiv:2506.08764.

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

Angela Fan, Edouard Grave, and Armand Joulin. 2020a. Reducing transformer depth on demand with structured dropout. *ICLR*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020b. Reducing transformer depth on demand with structured dropout. In *ICLR*.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Andreas Griewank and Andrea Walther. 2008. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2025. The unreasonable ineffectiveness of the deeper layers. *ICLR*.

Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. 2025. Slimllm: Accurate structured pruning for large language models. *ICML*.

Song Han, Huizi Pool, John Tran, and William Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*.

Babak Hassibi and David G Stork. 1992. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*.

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision*.

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. 2019. Robust learning with jacobian regularization. *Preprint*, arXiv:1908.02729.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

Michael F Hutchinson. 1989. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076.

Jiyeon Kim, Hyunji Lee, Hyowon Cho, Joel Jang, Hyeonbin Hwang, Seungpil Won, Youbin Ahn, Dohaeng Lee, and Minjoon Seo. 2025. Knowledge entropy decay during language model pretraining hinders new knowledge acquisition. *ICLR*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *EMNLP*.

Yann LeCun, John S Denker, and Sara A Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets. In *ICLR*.

Gui Ling, Ziyang Wang, and Qingwen Liu. 2024. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yao Lu, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu. 2024a. Reassessing layer pruning in llms: New insights and methods. *arXiv preprint arXiv:2411.15558*.

Yao Lu, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu. 2024b. Reassessing layer pruning in llms: New insights and methods. *arXiv preprint arXiv:2411.15558*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *ICLR*.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017a. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017b. Pruning convolutional neural networks for resource efficient inference. In *ICLR*.

Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. 2018. On the importance of single directions for generalization. In *ICLR*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. 2018. Sensitivity and generalization in neural networks: an empirical study. In *ICLR*.

Vardan Papyan. 2019. Measurements of three-level hierarchical structure in the outliers in the jacobians of deep neural networks. In *ICML*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *NAACL*, pages 2080–2094.

Barak A Pearlmutter. 1994. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. 2017. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in Neural Information Processing Systems*.

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 preprint arXiv:1910.10683*.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*.

Samuel Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. 2017. Deep information propagation. In *International Conference on Learning Representations*.

Shoaib Ahmed Siddiqui, Xin Dong, Greg Heinrich, Thomas Breuel, Jan Kautz, David Krueger, and Pavlo Molchanov. 2024. A deeper look at depth pruning of llms. *arXiv preprint arXiv:2407.16286*.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. 2017. Generalization error of invariant classifiers. In *International Conference on Artificial Intelligence and Statistics*.

Mohammadreza Soltani, Suya Wu, Yuerong Li, Jie Ding, and Vahid Tarokh. 2022. On the energy statistics of feature maps in pruning of neural networks with skip-connections. In *DCC 2022*, page 482. IEEE.

James Stewart, Dan Clegg, and Saleem Watson. 2021. *Calculus: early transcendentals*. Cengage.

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*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT*, pages 4149–4158.

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

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2021. Tuning large neural networks via zero-shot hyperparameter transfer. In *NeurIPS*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. LaCo: Large language model pruning via layer collapse. In *EMNLP 2024*, pages 6401–6417.

Shunshi Zhang and Bradly C. Stadie. 2020. One-shot pruning of recurrent neural networks by jacobian spectrum evaluation. In *International Conference on Learning Representations*.

A Jacobian Deviation as a Pruning Criterion

To motivate *Jacobian Deviation from Identity (JD)* as a pruning signal, we formalize how it relates to the expected loss increase when replacing a Transformer block by the identity map.

A.1 Pruning as Identity Replacement

Let the network be

$$F(x) = F_{>l}\big(f_l(h_l)\big), \qquad h_l = F_{<l}(x), \quad (18)$$

where f_l is the block under consideration. Pruning corresponds to

$$\tilde{F}(x) = F_{>l}(h_l), \quad (19)$$

i.e., replacing f_l with the identity. The block Jacobian is

$$J_l = \frac{\partial f_l(h_l)}{\partial h_l}. \quad (20)$$

For residual blocks $f_l(h) = h + G_l(h)$, we have $J_l = I + J_{G_l}$, and pruning enforces $G_l \equiv 0$, so the perturbation is captured by $J_l - I$.

A.2 Expected Loss Change under Pruning

Let $y = f_l(h_l)$ and define the pruning error $e = y - h_l$. Locally,

$$e \approx (J_l - I)\, h_l. \tag{21}$$

If LayerNorm approximately whitens h_l so that $\mathbb{E}[h_l] = 0$ and $\mathrm{Cov}(h_l) \approx \sigma^2 I$, then

$$\begin{aligned} \mathbb{E}\|e\|^2 &\approx \mathbb{E}\|(J_l - I)h_l\|^2 && (22)\\ &= \mathrm{tr}\big((J_l - I)\,\mathrm{Cov}(h_l)\,(J_l - I)^\top\big) && (23)\\ &\approx \sigma^2 \|J_l - I\|_F^2. && (24) \end{aligned}$$

Thus the expected representation error scales with the Frobenius deviation $\|J_l - I\|_F^2$.

Downstream, $F_{>l}$ propagates this perturbation to the loss. By a Lipschitz or Taylor bound on $F_{>l}$ and the loss $\mathcal{L}$,

$$|\Delta\mathcal{L}| = \big|\mathcal{L}(F_{>l}(y)) - \mathcal{L}(F_{>l}(h_l))\big| \;\lesssim\; C\|e\|, \tag{25}$$

for some local constant C. Taking expectations yields

$$\mathbb{E}[\Delta\mathcal{L}^2] \;\lesssim\; C^2\,\mathbb{E}\|e\|^2 \;\propto\; \|J_l - I\|_F^2. \tag{26}$$

Hence, ranking layers by

$$\mathrm{JD}(l) = \frac{1}{D}\|J_l - I\|_F^2, \qquad D = \dim(h_l), \tag{27}$$

directly targets the quantity governing expected pruning damage.

A.3 Efficient Estimation and Transformer Justification

Direct computation of $\|J_l - I\|_F^2$ is expensive. Using

$$\|J - I\|_F^2 = \|J\|_F^2 - 2\,\mathrm{tr}(J) + D, \tag{28}$$

we can apply Hutchinson probes $v \sim \mathcal{N}(0, I)$ (or Rademacher) for unbiased estimates:

$$\mathbb{E}\|J^\top v\|^2 = \|J\|_F^2, \quad \mathbb{E}[v^\top J v] = \mathrm{tr}(J). \tag{29}$$

These quantities are efficiently obtained with Jacobian-Vector products (JVP).

This formulation is particularly natural for Transformers: (i) residual blocks satisfy $J - I = J_G$, so JD measures the residual's derivative strength, which vanishes if the block is removed; (ii) LayerNorm encourages isotropic, zero-mean states, justifying the covariance simplification; (iii) unlike abstract importance scores, JD explicitly quantifies the risk of replacing a block by identity, aligning precisely with the pruning operation.

Experiment	Runtime
FD (RoBERTa, GLUE)	2h 55m
AD (RoBERTa, GLUE)	2h 40m
FD Pruning (T5-base, CoT)	1h 55m
AD Pruning (T5-base, CoT)	1h 40m
KE (RoBERTa, GLUE)	2h 35m
KE (T5, CoT)	1h 20m
ED Pruning (RoBERTa, GLUE)	2h 30m
LayerDropout (RoBERTa, GLUE)	1h 50m
LayerDropout (T5-base, CoT)	1h 10m
SlimLLM (RoBERTa, GLUE)	2h 05m
SlimLLM (T5, CoT)	1h 12m

Table 3: Runtime of key experiments on a Colab A100 GPU.

B Computing Infrastructure and Runtime

All experiments are performed within a Google Colab environment equipped with an NVIDIA A100 GPU, providing a standardized computational setting for JD assessment and layer-pruning studies on RoBERTa-base and T5-base across a suite of NLP benchmarks. Both architectures are initialized from publicly available pretrained checkpoints. For fine-tuning, we utilize a subset of twelve datasets, retaining up to 10,000 randomly selected training instances per dataset, or the full training set when fewer samples are available. Evaluation is conducted on a held-out portion, where 2,000 test examples are drawn at random from each dataset.

Table 3 reports the runtimes of the main experiments on a Colab A100 GPU, showing that both the model choice and the pruning method substantially affect wall-clock time.

C Use of Large Language Models

During the development of this manuscript, we used ChatGPT 5 as an auxiliary tool for both editing and coding support. For writing, it was used to improve grammar, spelling, and sentence-level clarity in selected parts of the paper. For experiments, it was used to help identify and resolve implementation issues in our Python code.