

TraceNAS: Zero-shot Search for LLM Pruning via Gradient Trace Correlation

Anonymous authors

Paper under double-blind review

Abstract

1 Structured pruning is essential for efficient deployment of Large Language
 2 Models (LLMs). The varying sensitivity of LLM sub-blocks to pruning
 3 necessitates the identification of optimal non-uniformly pruned models.
 4 Existing methods evaluate the importance of layers, attention heads, or
 5 weight channels in isolation. Such localized focus ignores the complex
 6 global structural dependencies that exist across the model. Training-aware
 7 structured pruning addresses global dependencies, but its computational
 8 cost can be just as expensive as post-pruning training. To alleviate the com-
 9 putational burden of training-aware pruning and capture global structural
 10 dependencies, we propose TraceNAS, a training-free Neural Architecture
 11 Search (NAS) framework that jointly explores structured pruning of LLM
 12 depth and width. TraceNAS identifies pruned models that maintain a high
 13 degree of loss landscape alignment with the pretrained model using a scale-
 14 invariant zero-shot proxy, effectively selecting models that exhibit maximal
 15 performance potential during post-pruning training. TraceNAS is highly
 16 efficient, enabling high-fidelity discovery of pruned models on a single
 17 GPU in 8.5 hours, yielding up to 10 $\times$ reduction in GPU-hours compared
 18 to training-aware methods. Evaluations on the Llama and Qwen families
 19 demonstrate that TraceNAS is competitive with training-aware baselines
 20 across commonsense and reasoning benchmarks.

21 1 Introduction

22 Recent breakthroughs in Large Language Models (LLMs) (Anthropic, 2023; OpenAI, 2023)
 23 have delivered unprecedented capabilities but require massive scale in parameters and
 24 computation. Foundational scaling laws (Kaplan et al., 2020) have established that smaller
 25 LMs are less sample-efficient, and thus training them from scratch is a resource-intensive
 26 bottleneck. Model compression enables the realization of efficient models that preserve
 27 the extensive knowledge of the pretrained base. Techniques such as quantization (Frantar
 28 et al., 2022; Liu et al., 2024), distillation (Li & Jin, 2022; Bercovich et al., 2025; Sarah et al.,
 29 2024) and pruning (Xia et al., 2023; Sun et al., 2023) have become essential, allowing for
 30 the accessible deployment of high-performance architectures. Among these, structured
 31 pruning (Ma et al., 2023) aims to remove architectural units like attention heads, weight
 32 columns, or transformer blocks to achieve immediate hardware speedup. This process
 33 can be naturally formulated as a Neural Architecture Search (NAS) (Sieberling et al.; Tao
 34 et al., 2023; Tang et al., 2025) problem, where the pretrained LLM is conceptualized as a
 35 super-network (Yu et al., 2020; Cai et al., 2019) that serves as the search space.

36 Structured pruning is inherently more disruptive than the localized masking used in un-
 37 structured pruning (Sun et al., 2023; Frantar & Alistarh, 2023). While unstructured methods
 38 leave the model’s architecture intact, structured pruning physically alters it and risks sever-
 39 ing critical pathways in the representational flow. The primary bottleneck in adapting
 40 unstructured methods to structured pruning is the lack of a comparable global metric across
 41 the model’s layers (Ma et al., 2023). Local importance heuristics such as activation-weighted
 42 magnitudes (Sun et al., 2023) or Hessian-based reconstructions (Frantar & Alistarh, 2023)
 43 identify redundancy within isolated layers and are often incomparable across model depth
 44 due to varying magnitude scales (Ma et al., 2023). We address this limitation by utilizing

45 gradients (Das et al., 2023; Yang et al., 2025) as global signals that naturally capture the
46 complex inter-dependencies across the model.

47 Previous search methods for structured
48 pruning employ mask learning (Xia et al.,
49 2023; Yuan et al., 2025) or training-aware
50 search (Tang et al., 2025; Bercovich et al.,
51 2025). While these training-aware ap-
52 proaches define the state-of-the-art, they
53 demand substantial computational re-
54 sources. This overhead stems from the
55 iterative gradient updates required for
56 *training-aware* model discovery and also
57 necessitates a continuous stream of un-
58 seen tokens to prevent pruned models
59 from overfitting to calibration data. This
60 cost can make the search for an effi-
61 cient model just as demanding as post-
62 pruning recovery, motivating the need
63 for *training-free* NAS (Wu et al., 2021; Tran
64 et al., 2021; Ingolfsson et al., 2022) us-
65 ing zero-shot performance proxies. How-
66 ever, existing proxies are primarily de-
67 signed to identify trainable, randomly
68 initialized architectures by focusing on
69 heuristics such as gradient stability (Abdelfattah et al., 2021) or loss landscape smooth-
70 ness (Li et al., 2023). These metrics are insufficient for LLM pruning as they prioritize
71 trainability, the ability of a network to learn from scratch. In contrast, the performance of
72 pruned LLMs depends on inheritance, the ability of the pruned model to recover perfor-
73 mance by preserving the pretrained loss landscape (Frankle & Carbin, 2018; Frankle et al.,
74 2020; Chen et al., 2020).

75 In this paper, we introduce TraceNAS, a training-free NAS framework for joint depth
76 and width pruning of LLMs. Our approach leverages the observation that a pretrained
77 model resides in broad, stable regions of the loss landscape (Frankle et al., 2020; Li et al.,
78 2018). We propose a zero-shot proxy that utilizes gradient trace to capture the impact of
79 structural pruning on model sensitivity. Specifically, the proxy measures the retention-
80 weighted aggregate over the Pearson Correlation coefficients between the gradient traces
81 of the pruned and pretrained base models. Since structured pruning alters activation and
82 gradient magnitudes, the use of Pearson Correlation ensures scale invariance, capturing the
83 alignment of gradient traces on the pretrained loss landscape independent of scale shifts.

84 To ensure the search remains tractable across varying model scales, we compute these
85 gradients within a low-rank (Hu et al., 2022; Zhao et al., 2024) subspace. By leveraging
86 the intrinsic low-dimensionality (Aghajanyan et al., 2021) of LLMs, TraceNAS captures the
87 principal gradient components of the model while reducing memory overhead by orders of
88 magnitude. Unlike training-aware methods which require significant memory overhead
89 to maintain optimizer states and activations for backpropagation across a large set of
90 calibration tokens, our training-free approach only requires the gradients of each candidate
91 computed on a minimal calibration set. This reduction in memory consumption enables
92 high-fidelity model discovery (Fig. 1), bypassing the compute intensive requirements typical
93 of training-aware NAS. The primary contributions of this work are:

- 94 1. **Scalable Training-free NAS Framework:** We introduce TraceNAS, a unified pruning
95 framework (Fig. 2) for the joint optimization of LLM depth and width. Our approach
96 enables the zero-shot discovery of non-uniform pruned models, facilitating efficient
97 compression across model scales without search-time training.

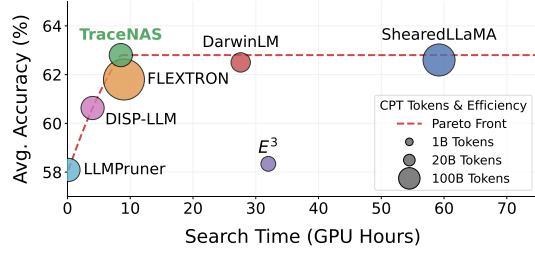


Figure 1: **Search Efficiency.** TraceNAS lies on the Pareto front of the accuracy-efficiency trade-off, identifying high-potential architectures in 8.5 GPU hours¹. We achieve up to 10× reduction in search time in comparison to competitive base-lines. The area of the bubble is proportional to the total tokens used for recovery training, highlighting that TraceNAS identifies architectures that have high recovery potential.

¹TraceNAS, ShearedLlama, Flextron, DISP-LLM and LLMPruner report search time on NVIDIA A100, DarwinLM on L40S and E³ reports NPU hours. Accuracy values reported in the figure are for Llama2-7B.

2. **Zero-shot Proxy for Functional Inheritance:** We introduce a scale-invariant, gradient-based zero-shot proxy that evaluates gradient alignment between the pruned and pretrained models. Our proxy achieves superior Spearman Rho (Spearman, 1904) ($\rho = 0.94$) and Kendall Tau (Kendall, 1938) ($\tau = 0.82$) correlations with downstream accuracy, enabling accurate zero-shot discovery.
3. **Efficient Architecture Discovery:** We show that TraceNAS achieves up to $10\times$ reduction in both GPU-hours and total calibration data compared to training-aware baselines (Fig. 1). This drastically lowers the overhead for large-scale model compression by ensuring search costs are significantly lower than recovery training.

2 Related Work

2.1 Language Model (LM) Pruning

LLM pruning methods vary by granularity: unstructured weight sparsification (Frantar & Alistarh, 2023) or structured removal (Ma et al., 2023; Kim et al., 2024) of transformer blocks, attention heads or weight channels. These methods are further categorized by their computational complexity relative to the model’s hidden dimension d . Early $O(d^2)$ magnitude pruning (Han et al., 2015) methods evaluate weights in isolation, ignoring the outlier features characteristic of LLMs (Kovaleva et al., 2021; Luo et al., 2021; Yin et al., 2023). To preserve these outliers, $O(d^2)$ metrics like Wanda (Sun et al., 2023) and FLAP (An et al., 2024) utilize weight-activation products. Furthermore, $O(d^3)$ optimization-based methods like SparseGPT (Frantar & Alistarh, 2023) and LLM-Pruner (Ma et al., 2023) utilize second-order inverse Hessians (Hassibi et al., 1993) to minimize layer-wise reconstruction error. However, these local heuristics prioritize layer properties and fail to account for how pruning-induced errors propagate through the model.

2.2 Influence-based Importance

Beyond these heuristics, recent research has explored influence functions (IF) (Koh & Liang, 2017; Kwon et al., 2023) to better capture how model perturbations affect sensitivity. For instance, LayerIF (Askari et al., 2025) uses IF to estimate layer importance; however, it remains locally heuristic as it perturbs layers individually to measure validation loss sensitivity. This inherently fails to capture the joint impact of multi-layer and width pruning on the model’s representation. Furthermore, the $O(d^3)$ complexity of standard IF is prohibitive for LLMs, requiring inverse Hessian computations for all model parameters. In contrast, TraceNAS adopts an efficient first-order gradient-tracing approach inspired by TraIn (Pruthi et al., 2020) to evaluate the global impact of pruning. By operating with $O(d^2)$ complexity, TraceNAS identifies non-uniform pruned architectures that maintain functional alignment with the pretrained base model, bypassing the computational bottlenecks and local limitations of LayerIF. We provide a detailed comparison between IF, IF-based pruning and TraceNAS in Appendix D.

2.3 Training-Aware LM Pruning

Recent structured pruning methods, such as ShearedLlama (Xia et al., 2023) and E³ (Yuan et al., 2025), use mask learning via L_0 regularization and differential mask optimization. While effective, these methods target uniform sparsity across layers, failing to account for the heterogeneous importance distribution across a model’s width (Tao et al., 2023). This has prompted a shift toward discovering non-uniform architectures via evolution search. Methods like SIMPLE (Tao et al., 2023) and EvoPress (Sieberling et al.) employ evolutionary optimization for heterogeneous discovery. On the other hand, DarwinLM (Tang et al., 2025) uses a curriculum-style training-aware search, performing lightweight fine-tuning for every searched candidate. Similarly, PUZZLE (Bercovich et al., 2025) utilizes distillation-based NAS to identify optimal configurations by minimizing block-wise KL divergence. However, the training-aware search inherent to these methods mean their model discovery can be just as computationally intensive as the post-pruning recovery training.

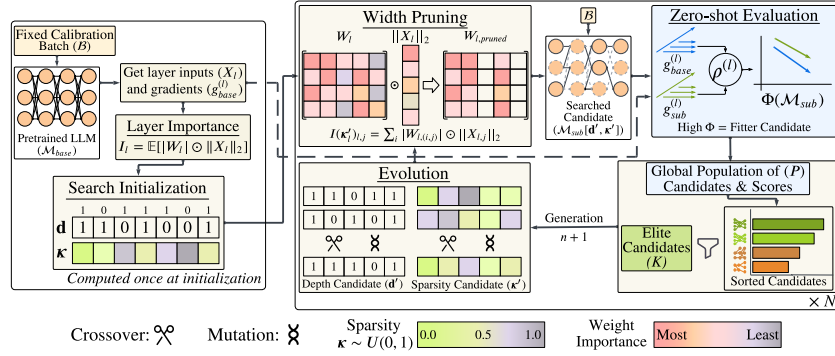


Figure 2: **Visualization of the TraceNAS search framework.** TraceNAS uses a gradient-based, training-free proxy to guide structural pruning. Following a one-time initialization of base gradient traces (g_{base}) and importance scores (I_l), a population of depth (d) and width (κ) candidates undergoes iterative evolution via crossover and mutation. Each width configuration is realized using an $O(d^2)$ activation-weighted heuristic. Subsequently, candidates $\mathcal{M}_{sub}$ are ranked by the zero-shot proxy Φ , which measures the gradient trace alignment between the active layers of $\mathcal{M}_{sub}$ relative to $\mathcal{M}_{base}$.

2.4 Zero-shot NAS for LM Pruning

As models scale, these training-intensive searches become intractable, necessitating training-free solutions for NAS. Existing proxies are formulated to identify trainable, randomly initialized architectures by focusing on heuristics such as gradient stability (Abdelfattah et al., 2021), model expressivity (Mellor et al., 2021; Jiang et al., 2023), or loss landscape smoothness (Li et al., 2023). These proxies are insufficient for evaluating pruned models as they do not account for how well a model inherits pretrained knowledge. To address this, recent works LPZero (Dong et al., 2024a) and Pruner-Zero (Dong et al., 2024b) formulate zero-shot pruned model discovery as a two-fold problem: first searching for unique symbolic pruning metrics for the target pretrained model, and subsequently applying these metrics to discover the optimal structured (Dong et al., 2024a) or unstructured (Dong et al., 2024b) sub-network. While modular, the dual-stage process adds overhead to the search pipeline. In contrast, TraceNAS introduces a unified training-free fitness proxy that directly evaluates a compressed model’s functional alignment with its pretrained base, streamlining the search for pruned models that have high performance potential and fast convergence during post-pruning recovery.

3 Methodology

3.1 Problem Definition

We formulate pruned model discovery as a constrained discrete search over a pretrained super-network, $\mathcal{M}_{base}$ (Cai et al., 2019; Yu et al., 2020). We seek an optimal $\hat{\mathcal{M}}_{sub} \in \mathcal{G}$ that maximizes a training-free proxy $\Phi(\cdot, \cdot)$, that measures functional inheritance; the ability of $\mathcal{M}_{sub}$ to recover performance by preserving the gradient direction of $\mathcal{M}_{base}$ on its loss landscape:

$$\hat{\mathcal{M}}_{sub} = \arg \max_{\mathcal{M} \in \mathcal{G}} \Phi(\mathcal{M}_{sub}, \mathcal{M}_{base}) \text{ s.t. } \mathcal{P}(\mathcal{M}_{sub}) \leq C,$$

where $\mathcal{G}$ is the search space, $\mathcal{P}(\cdot)$ denotes the parameter count, and C is the target constraint. By maximizing Φ , we identify models with maximal potential for performance recovery during post-pruning *continued pretraining* (CPT) without the overhead of search-time training.

3.2 Search Space Encoding

The search space $\mathcal{G}$ is defined as a joint distribution over the model’s depth and sub-block-wise width, enabling the discovery of non-uniformly pruned architectures from $\mathcal{M}_{base}$. By

operating at sub-block granularity, the TraceNAS search framework treats attention and MLP modules as independent units, thereby capturing diverse structural sparsity.

Architecture Encoding A candidate $\mathcal{M}_{sub}$ is parameterized by a tuple $(\mathbf{d}, \kappa)$. The depth configuration $\mathbf{d} \in \{0, 1\}^L$ is a discrete mask over the L blocks of $\mathcal{M}_{base}$. Setting $d_l = 0$ deactivates the l -th block while preserving the residual stream, allowing the search to explore varying model depths. We enforce a minimum depth constraint, $\sum_{l=1}^L d_l \geq L_{min}$, to ensure sufficient computational capacity for sequential reasoning. Simultaneously, the width configuration $\kappa = \{\kappa_1, \dots, \kappa_L\}$ where $\kappa_l = (r_{attn}^{(l)}, r_{mlp}^{(l)}) \in (0, 1]^2$ defines the sparsity ratio for each attention and MLP sub-block. By decoupling the sparsity ratios, our framework identifies heterogeneous architectures that distinctly prioritize retention in sensitive sub-modules while respecting the global budget C .

3.3 In-Place Architectural Realization

We realize candidate sub-networks $\mathcal{M}_{sub}$ by mapping sparsity ratios κ to binary masks and implementing them via an efficient in-place masking strategy.

Width Mask Generation To realize a layer’s width configuration κ_l into a weight mask, we use the activation-weighted product: $I(\kappa_l)_{l,j} = \sum_i |W_{l,(ij)}| \cdot \|X_{l,j}\|_2$ (Sun et al., 2023) as a saliency metric. This approach explicitly accounts for the outlier features characteristic of LLMs (Kovaleva et al., 2021; Luo et al., 2021; Yin et al., 2023) by scaling weight magnitudes by the L_2 -norm of their corresponding input activations. By identifying channels that are functionally critical to the model’s representational flow, $I(\kappa_l)_{l,j}$ ensures that mask realizations are tailored to preserve the underlying representation of $\mathcal{M}_{base}$.

Adopting this $O(d^2)$ heuristic avoids the $O(d^3)$ bottleneck of second-order reconstruction methods (Frantar & Alistarh, 2023; Ma et al., 2023; Tang et al., 2025), enabling the high-frequency scoring and realization of hundreds of distinct candidates during the evolutionary search. Crucially, while $I(\kappa_l)_{l,j}$ provides a local channel-selection heuristic, the global distribution of the sparsity ratios κ_l is governed by the TraceNAS proxy Φ (Sec. 3.4), which accounts for the inter-layer dependencies and impact of pruning that local heuristics ignore.

In-Place Masking To score candidates, we use an in-place masking strategy operating directly on the weight matrices of $\mathcal{M}_{base}$. For depth, $\mathcal{M}_{base}$ is modified according to $d_l \in \mathbf{d}$; if $d_l = 0$, the block is bypassed and activations are routed through the residual connection, ensuring functional continuity (Veit et al., 2016; Gromov et al., 2024). For active blocks ($d_l = 1$) the binary masks are applied via a modify-then-restore pointer: $W'_{base} = W_{base} \odot \text{Mask}(\kappa_l)$. This allows the search framework to execute forward and backward passes over a calibration set to capture the cumulative effects of pruning without the memory overhead of instantiating each $\mathcal{M}_{sub}$ in the population. Once the gradient trace is recorded for scoring, the original weights are restored, preserving $\mathcal{M}_{base}$ for subsequent search iterations.

3.4 TraceNAS: Evaluating Functional Alignment

To evaluate the functional integrity and recovery potential of candidate models $\mathcal{M}_{sub}$, we propose the proxy Φ . While structured pruning induces immediate representational shifts (Tran et al., 2022), local importance-based metrics (Sun et al., 2023; An et al., 2024) often overlook inter-layer dependencies that are inherently captured by gradients (Das et al., 2023). To address this, we define Φ as a zero-shot, gradient-based proxy that quantifies functional inheritance. Φ measures the directional alignment between the gradient trace of $\mathcal{M}_{sub}$ and $\mathcal{M}_{base}$. This directional alignment allows Φ to identify models that are favorably positioned within the pretrained convergence basin, thereby enhancing their potential for functional recovery.

Establishing the Functional Anchor We define the optimal functional state as the gradient trace g_{base} of the pretrained model $\mathcal{M}_{base}$. Using a calibration set $\mathcal{B}$, g_{base} provides the directional anchor within the optimization landscape that sub-networks $\mathcal{M}_{sub}$ must align with to ensure functional inheritance. The gradient traces for the base and candidate models are calculated as $g = \mathbb{E}_{b \in \mathcal{B}} [\nabla_{\theta} \mathcal{L}(\mathcal{M}(b; \theta))]$ where θ denotes the trainable model parameters.

Low-Rank Gradient Manifold To ensure computationally tractability across model sizes, we compute gradient traces g within a low-rank subspace. Since storing full-rank LLM gradients is memory-prohibitive, we attach Low-Rank Adapters (LoRA) (Hu et al., 2022) to all linear projections. This approach leverages the observation that the functional knowledge of LLMs resides within a low-dimensional manifold (Aghajanyan et al., 2021; Zhao et al., 2024), allowing us to capture global dependencies while significantly reducing the memory footprint of gradient storage.

Measuring Functional Alignment To quantify how effectively $\mathcal{M}_{sub}$ inherits the functional state of $\mathcal{M}_{base}$, we calculate the Pearson Correlation Coefficient $\rho^{(l)}$ per layer $l \in L$. For each transformer-block l , $\rho^{(l)}$ measures the alignment between the low-rank gradient traces of the pruned attention and MLP sub-blocks:

$$\rho^{(l)} = \frac{1}{N_l} \left\langle \left(\frac{g_{sub}^{(l)} - \mu_{g_{sub}^{(l)}}}{\sigma_{g_{sub}^{(l)}}} \right), \left(\frac{g_{base}^{(l)} - \mu_{g_{base}^{(l)}}}{\sigma_{g_{base}^{(l)}}} \right) \right\rangle \quad (1)$$

where N_l is the dimensionality of the low-rank subspace, and μ, σ denote the mean and standard deviation of the gradient elements. By standardizing the traces, we decouple the directional alignment from magnitude shifts inherently induced by structural pruning.

Retention-Weighted Aggregation We formulate the final proxy Φ by aggregating sub-block correlations using a retention-weighted summation. While gradient standardization ensures local scale-invariance, a uniform average of $\rho^{(l)}$ across layers fails to account for the heterogeneous representation capacity of $\mathcal{M}_{sub}$. We therefore weight each correlation coefficient by its corresponding layer retention ratio $\bar{r}^{(l)} = 1 - r^{(l)} \in \kappa_l$:

$$\Phi(\mathcal{M}_{sub}, \mathcal{M}_{base}) = \sum_{l \in \text{Attn}} \bar{r}_{attn}^{(l)} \cdot \rho^{(l)} + \sum_{l \in \text{MLP}} \bar{r}_{mlp}^{(l)} \cdot \rho^{(l)} \quad (2)$$

This formulation anchors the global score in the high-capacity regions of $\mathcal{M}_{sub}$, prioritizing the sub-blocks that serve as the primary repositories of functional inheritance. By weighting $\rho^{(l)}$ by parameter density, we prevent Φ from being skewed by the high-variance noise typical of aggressively pruned sub-blocks.

3.5 Selection & Evolutionary Mechanics

Search Space Evolution We navigate $\mathcal{G}$ via an evolutionary strategy driven by Φ as shown in Fig. 2. The population P_0 is warm-started using a global importance prior (Appendix C). A hybrid strategy optimizes the joint $(\mathbf{d}, \kappa)$ space: depth evolution utilizes contiguous-range crossover and bit-flip mutations, while width evolution employs interpolation crossover and multiplicative Gaussian jitter. This joint optimization explores architectural trade-offs between depth and width while adhering to the parameter constraint C.

Search Granularity Activations at the attention output (W_o) and MLP down (W_{down}) projection layers serve as importance proxies for upstream weights. We apply the resulting masks to W_q, W_k, W_v, W_o for attention and $W_{up}, W_{gate}, W_{down}$ for MLP. For Grouped-Query Attention (GQA) (Ainslie et al., 2023) models, W_k and W_v are preserved to maintain representation integrity. W_o importance scores are aggregated head-wise to ensure that the number of pruned channels remains a multiple of the head dimension d_{head} , preserving the structural integrity required for multi-head and grouped-query attention. Finally, pruned MLP hidden dimensions are rounded to multiples of $m = 32$ to maintain efficient tensor operations.

3.6 Interpreting Functional Inheritance through Gradient Trace Alignment

The effectiveness of Φ as a proxy for functional inheritance is driven by the following observations:

- **Manifold Anchoring:** High-performing pretrained models reside within flat, stable convergence basins (Frankle et al., 2020; Li et al., 2018). A high alignment score Φ implies that the first-order Taylor expansion of the sub-network’s loss surface remains

congruent with that of the base model. By preserving the gradient’s direction, $\mathcal{M}_{sub}$ remains anchored within the original functional manifold. This ensures that post-pruning recovery initiates from a region of high directional certainty, enabling a stable loss trajectory rather than necessitating a costly search for a new local minimum.

- **Global Sensitivity via Chain-Rule:** Unlike metrics that evaluate layers in isolation (Sun et al., 2023; Askari et al., 2025), g_{sub} is computed via backpropagation through the masked computational graph. This captures inter-layer disruptions: if an upstream block bottlenecks the representational flow, subsequent layer gradient traces will de-correlate from the base model optimization manifold. This causes a sharp drop in Φ , allowing the proxy to detect structural incoherence and broken residual streams that local heuristics are inherently blind to.
- **Magnitude Invariance:** By decoupling the proxy from absolute gradient magnitudes, Φ prevents the search from being biased by model size and focuses on functional preservation. This design reflects the use of AdamW (Loshchilov & Hutter, 2017), where the update $\Delta w_t \propto \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ normalizes scale to prioritize directional consistency over raw magnitude. Furthermore, Pre-LayerNorm and residual connections stabilize activation scales across the depth of the model, ensuring that Φ captures the underlying optimization trajectory rather than artifacts of the pruning mask.

4 Experiments

4.1 Experimental Setup

Models and Datasets We evaluate TraceNAS across multiple scales using Llama-2-7B (Touvron et al., 2023), Llama-3.1-8B (Dubey et al., 2024) and Qwen-2.5-14B (Team, 2024b; Yang et al., 2024), covering both Multi-Head Attention (MHA) (Vaswani et al., 2017) and Grouped-Query Attention (GQA) (Ainslie et al., 2023) architectures. We use the FineWeb-Edu (Penedo et al., 2024) 100BT dataset for both evolutionary search and post-pruning CPT. For the search, we use a fixed calibration set of 65,536 tokens (16 sequences) per pruned candidate to evaluate a total population of 1500 candidates. This is followed by 20B tokens of post-pruning CPT. Results on additional models (Dubey et al., 2024; Team, 2025; 2024a; Biderman et al., 2023) and CPT token budgets is provided in Appendix F.3 and F.4.

Implementation Details The evolutionary search is performed with a population of 30 candidates over 50 search iterations. On a single NVIDIA H200 GPU, the search concludes in ~ 2 hours, a significant acceleration over the 8.5-hour A100 baseline reported in Fig. 1. The highest scored sub-networks then undergo CPT on a cluster of six H200 GPUs (~ 16 hours) using a global batch size of 1024 and peak learning rate of $1e^{-4}$. We use a Warmup-Stable-Decay (WSD) (Hu et al., 2024) scheduler, with context lengths set to 4096 for Llama-2 and Qwen-2.5, and 8192 for Llama-3.1. Detailed hyperparameters are available in Appendix C.5.

Baselines We benchmark TraceNAS against leading structured and layer-pruning methods. This includes, mask-learning methods, ShearedLLaMA (Xia et al., 2023) and E^3 -Pruner (Yuan et al., 2025), which employ Lagrangian optimization. We compare performance with LLMPruner (Ma et al., 2023) which uses Taylor series approximation of the gradient as the importance metric for pruning. For training-aware pruning, we compare against DarwinLM (Tang et al., 2025), LoRAP (Li et al., 2024) which incorporates fine-tuning with LoRA (Hu et al., 2022) during pruning and DISP-LLM (Gao et al., 2024b) which trains a hypernetwork for mask generation. We further evaluate against Minitron (Sreenivas et al., 2024), Flextron (Cai et al., 2024), and uniformly pruned models found via TraceNAS to isolate the benefits of non-uniform pruning. Comparisons with unstructured pruning methods like Wanda (Sun et al., 2023), FLAP (An et al., 2024) and PrunerZero (Dong et al., 2024b) and LoRA finetuned models are included in Appendix F.5.

Evaluation Benchmarks Models are evaluated using lm-evaluation-harness (Gao et al., 2024a) across 0-shot commonsense reasoning benchmarks: ARC-easy (Clark et al., 2018), LogiQA (Liu et al., 2020), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017) and BoolQ (Clark et al., 2019). We also report 5-shot performance on MMLU (Hendrycks et al., 2020) and WinoGrande (Sakaguchi et al., 2020), 10-shot on HellaSwag (Zellers et al., 2019), and 25-shot on ARC Challenge (Clark et al., 2018).

4.2 TraceNAS Proxy Validation: Performance Correlation and Stability

To evaluate TraceNAS as a proxy for recovery potential and verify hyperparameter consistency, we analyze 70 models sampled from the AmoebaLLM (Fu et al., 2024) search space. This evaluates the proxy in an environment unbiased by our evolutionary search and large-scale CPT setup. We compute Spearman Rho (SP) and Kendall Tau (KT) correlations against Wikitext-2 perplexity (PPL), MMLU accuracy and average downstream accuracy after Once-for-All (OFA) (Cai et al., 2019; Sukthanker et al.) finetuning on Alpaca (Taori et al., 2023).

Method	PPL		MMLU		Avg. Acc.	
	SP	KT	SP	KT	SP	KT
#Params	0.87	0.72	0.11	0.07	0.66	0.50
NASWOT	0.69	0.50	0.35	0.26	0.39	0.29
ZiCo	0.87	0.72	0.11	0.07	0.66	0.50
GradNorm	0.93	0.79	0.45	0.32	0.93	0.78
Synaptic	0.93	0.78	0.37	0.25	0.89	0.73
MeCo	0.92	0.79	0.03	0.02	0.75	0.58
PrunerZero	0.44	0.31	0.88	0.69	0.68	0.50
TraceNAS – Dot	0.57	0.46	0.35	0.25	0.58	0.47
TraceNAS – Cos	0.91	0.77	0.48	0.35	0.91	0.79
TraceNAS – Unw.	0.85	0.71	0.23	0.16	0.77	0.62
TraceNAS (Ours)	0.93	0.79	<u>0.54</u>	<u>0.39</u>	0.94	0.82

Table 1: Correlation of zero-shot proxies with model performance. Best in **bold**, second best underlined.

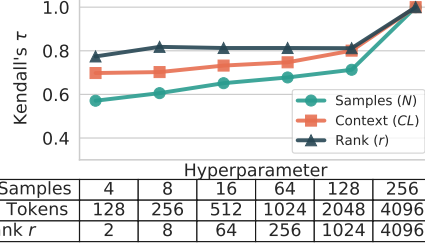


Figure 3: TraceNAS proxy stability. High τ values demonstrate that TraceNAS consistently ranks models across hyperparameters.

Correlation with Downstream Performance TraceNAS achieves superior ranking correlation with downstream performance, as shown in Table 1. Proxies like NASWOT (Mellor et al., 2021) and ZiCo (Li et al., 2023) rank models based on their expressivity and convergence capability from random initializations, in contrast TraceNAS explicitly accounts for pretrained knowledge inheritance. By measuring the alignment between the pruned and base model gradient traces, Φ captures how structural changes impact the functional sensitivity of the model. This allows the proxy to account for complex reasoning dependencies that feature-based or loss-landscape smoothness-based metrics fail to capture.

While GradNorm (Abdelfattah et al., 2021) and Synaptic Saliency (Tanaka et al., 2020) effectively rank PPL, they prioritize the health of the gradient signal and whether pruning disproportionately impacts individual layers. This provides an incomplete picture of the candidate’s representational integrity. Similarly, while MeCo (Jiang et al., 2023) performs well on PPL, it fails to translate to task-specific accuracy. PrunerZero (Dong et al., 2024b) achieves the highest correlation on MMLU by incorporating gradient information, but this success does not generalize to PPL or average downstream accuracy. Additionally, #Params has high ranking potential for both PPL and average accuracy, however this does not translate to a good measure of a model’s knowledge retention capability, which is needed for MMLU. This tells us that while increasing model size is a good indicator of generalizability potential, #Params cannot capture the nuances required to analyze a pruned model’s factual knowledge and reasoning capabilities.

In contrast, TraceNAS maintains superior ranking stability across all three metrics by leveraging Retention-Weighted Pearson Correlation (Φ). Φ is specifically designed to isolate functional signals from directional noise through gradient centering. By using layer-wise retention as a weighting factor, Φ anchors the global fitness score in high-capacity regions. This prevents the ranking from being skewed by the high variance noise typical of aggressively pruned sub-blocks. We further validate the choice of our proxy by comparing against alternative alignment metrics, including dot product, cosine similarity and unweighted aggregation of Pearson Correlation Coefficients. A detailed analysis of the correlation results is presented in Table 1 is provided in Appendix E.

Hyperparameter Stability Analysis We evaluate the robustness of TraceNAS by computing the Kendall τ correlation across search hyperparameters and their respective highest settings ($N = 256$, $CL = 4096$, $r = 4096$). As shown in Fig. 3, the proxy maintains high ranking consistency across all variables, confirming that gradient trace alignment effectively

captures model inheritance even under constrained search settings. Our analysis of these internal ranking correlations follows:

- **Number of Calibration Samples (N):** We observe that correlation monotonically align with the $N = 256$ configuration as sample density increases. While ranking sensitivity is present at $N = 4$ with $\tau \approx 0.57$, stability improves significantly to $\tau \approx 0.71$ at $N = 128$. This trend suggests that representative gradient alignment is captured with relatively few samples, justifying our sample-efficient search settings.
- **Context Length (CL):** We observe that correlation with $CL = 4096$ remains high across context lengths, rising from $\tau \approx 0.70$ at shorter contexts to $\tau \approx 0.80$ at $CL = 2048$. The high agreement for all $CL \geq 1024$ rankings suggests that the proxy’s saliency is largely preserved once the model captures sufficient long-range dependencies, allowing for significantly reduced compute during pruned model discovery.
- **Low-rank Gradient Subspace (r):** We observe that proxy stability is most significant across gradient subspace dimensions. Correlation with full-rank ($r = 4096$) configuration remains consistently high, exceeding $\tau \approx 0.77$ even at $r = 2$, and peaking at $\tau \approx 0.82$ for $r = 8$ before stabilizing. This validates our use of $r = 64$ during the search, confirming that low-rank gradient traces are sufficient to capture functional nuances while minimizing search memory overhead.

Detailed inter-hyperparameter correlations, comparing each configuration directly to downstream task performance, are provided in Appendix E.3.

4.3 Main Results

We evaluate TraceNAS for pruned model discovery by evaluating on Llama-2-7B, Llama-3.1-8B, and Qwen-2.5-14B Instruct to 2.7B, 4.6B, and 8.4B parameters, respectively. As shown in Tables 2 and 3, TraceNAS identifies sub-networks with high sample efficiency, requiring only 98M total search tokens, a $10\times$ and $4\times$ reduction compared to DarwinLM and ShearedLLaMA, while achieving superior accuracy. This validates the ranking capability and stability of our proxy, as observed in Sec. 4.2.

Table 2: Pruning results for LLaMA-2-7B. Averages are calculated across eight reasoning benchmarks.

Method	#Params	Srch	Recv	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama-2-7B	6.7B	–	–	77.69	74.11	76.34	52.81	78.96	93.80	29.80	77.73	70.16
LoRAP	2.7B	16K	50K steps	57.20	47.90	31.30	26.30	30.00	51.20	27.50	61.90	41.66
Flextron ²	3.4B	1K	90B	74.10	62.00	66.50	–	68.50	–	–	–	–
E ³	2.7B	0.5B	–	71.20	57.30	65.40	38.10	54.60	88.40	28.10	63.60	58.34
ShearedLLaMA	2.7B	0.4B	50B	75.80	64.20	67.00	41.20	70.80	90.80	28.20	63.00	62.63
DarwinLM	2.7B	1.0B	10B	73.12	63.77	69.94	43.94	66.95	90.20	28.57	64.10	62.57
Uniform	3.4B	98M	20B [†]	65.10	58.30	69.50	38.20	54.30	87.10	26.03	59.90	57.30
LLMPruner	2.86B	1.3K	20B [†]	68.43	59.52	65.52	36.84	59.73	88.71	26.76	59.21	58.09
DISP-LLM	2.7B	–	20B [†]	70.83	62.42	68.52	39.75	65.87	89.21	27.98	60.52	60.63
ShearedLLaMA	2.7B	0.4B	20B [†]	72.92	60.80	67.46	38.06	64.73	90.30	28.72	63.91	60.86
DarwinLM	2.7B	1.0B	20B [†]	73.17	63.59	69.90	42.91	66.57	89.00	29.18	63.63	62.24
TraceNAS (Ours)	2.7B	98M	20B	73.40	63.97	70.20	43.80	68.54	91.10	29.71	61.76	62.81

Performance on Llama-2-7B TraceNAS achieves the highest average accuracy on Llama-2 7B across eight benchmarks, outperforming all reported baselines. To ensure a fair comparison, we train DarwinLM, ShearedLLaMA, DISP-LLM, and LLMPruner on our 20B token distribution ([†]). Notably, ShearedLLaMA’s performance drops when restricted to our 20B token budget, highlighting its reliance on high-token-count recovery to reach competitive accuracy. In contrast, DarwinLM performs competitively at 10B tokens, leveraging search-time training to accelerate recovery. While TraceNAS requires 20B tokens to exceed these results, this recovery is justified by $3.5\times$ lower search overhead, thus enabling high-fidelity discovery. Additionally, LLMPruner does not surpass TraceNAS performance, and

[†] Models trained on our CPT data setup and the original codebase provided in the respective papers.

Table 3: Pruning results for Llama-3.1-8B and Qwen-2.5-14B-Instruct. Averages are calculated across the eight reasoning benchmarks, not including MMLU.

Method	#Params	Srch	Recv	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama-3.1-8B	8B	–	–	64.78	81.22	77.42	81.64	57.59	81.81	96.20	31.02	82.14	73.63
Minitron ²	4.4B	–	94B	60.50	76.82	73.50	76.89	56.14	76.03	96.20	31.02	83.14	71.22
DarwinLM	4.6B	1B	10B	43.21	74.59	65.03	73.27	51.27	71.13	93.40	30.72	71.00	66.30
Uniform (Ours)	4.6B	98M	20B [†]	28.76	68.32	58.54	68.23	47.89	66.42	88.48	27.54	66.59	61.50
LLMPruner	4.6B	1.3K	20B [†]	26.45	54.76	48.93	25.16	27.04	25.81	21.8	24.42	59.63	35.94
DarwinLM	4.6B	1B	20B [†]	43.32	74.53	65.26	73.13	51.62	71.17	93.24	30.76	70.08	66.22
TraceNAS (Ours)	4.6B	98M	20B	30.12	74.23	66.08	73.50	52.26	69.01	93.90	30.81	73.00	66.60
Qwen-2.5 14B	14B	–	–	82.53	82.10	79.71	85.81	72.44	85.10	96.70	41.16	87.95	78.87
DarwinLM	8.4B	1B	10B	55.97	78.12	70.63	79.41	57.42	74.93	89.30	33.10	73.57	69.56
E ³	8.2B	0.5B	–	36.50	76.90	63.00	76.00	47.90	67.00	93.70	30.00	66.50	65.13
Uniform (Ours)	8.4B	98M	20B [†]	51.23	71.32	62.64	77.12	50.02	69.89	88.91	28.53	69.45	64.73
LLMPruner	8.4B	1.3K	20B [†]	25.72	51.30	50.19	28.84	27.30	28.56	25.6	23.04	39.23	34.26
DarwinLM	8.4B	1B	20B [†]	56.97	78.17	72.09	79.86	57.03	74.96	90.32	32.80	73.17	69.80
TraceNAS (Ours)	8.4B	98M	20B	58.79	76.63	73.24	81.07	56.05	73.77	95.70	32.56	73.70	70.34

DISP-LLM requires hypernetwork training and post-pruning CPT, and still does not match TraceNAS performance and is a more expensive technique end-to-end.

Furthermore, TraceNAS significantly outperforms LoRAP (Li et al., 2024), E³-Pruner (Yuan et al., 2025), and our uniform baseline. Our 2.7B model has comparable performance to Flextron (Cai et al., 2024) despite the latter having a larger parameter footprint and requiring $4.5\times$ more CPT tokens. The substantial performance gap between TraceNAS and the uniform baseline (5.51% average) underscores the necessity of non-uniform architecture search for maintaining representational integrity. Additional results on pruned Llama-2, specifically zero-shot performance, baseline comparisons of model perplexity and performance scaling at 10B and 50B training tokens, in Appendices C.3, E.4 and F.4, respectively.

Generalization to GQA Architectures Evaluation on Llama-3.1 and Qwen-2.5 demonstrates the generalizability of TraceNAS to GQA architectures. For Llama-3.1-8B, TraceNAS significantly improves on the uniform baseline and outperforms DarwinLM across reasoning tasks such as ARC-C (52.26%) and BoolQ (73.00%). On Qwen-2.5-14B, TraceNAS maintains its lead with a 70.34% average, surpassing both DarwinLM (69.80%) and E³-Pruner (65.13%). Notably, LLM-Pruner (Ma et al., 2023) exhibits severe performance degradation on these architectures. We evaluate its original search strategy without modification for comparison.

While Minitron (Sreenivas et al., 2024) achieves higher accuracies across reasoning baselines, it requires an intensive 94B tokens for recovery, approximately $4.7\times$ the computational cost of TraceNAS. Across all architectures, TraceNAS produces sub-networks whose gradient traces are highly aligned with the pretrained gradient trace, allowing for high-fidelity recovery and discovered at a fraction of the computational search cost required by existing training-aware methods. Detailed pruned model performance before CPT, inference, search, and recovery speedup analysis against all publicly available baselines are provided in Appendix C.3, F.1, and F.2, respectively.

5 Conclusion

In this work, we introduce TraceNAS, a zero-shot NAS framework for non-uniform structured pruning of LLMs. By leveraging a novel scale-invariant gradient proxy, TraceNAS identifies sub-networks that maintain high gradient trace alignment with the pretrained base model. This alignment ensures the preservation of functional inheritance, allowing for efficient recovery during continued pretraining. Our framework demonstrates consistent performance gains across Llama-2, Llama-3.1, and Qwen-2.5 architectures, achieving an order-of-magnitude reduction in search overhead compared to training-aware baselines. By eliminating the prohibitive cost of search-time training, TraceNAS provides a scalable and high-fidelity foundation for model compression that effectively retains the complex reasoning capabilities of dense pretrained models.

²Flextron Cai et al. (2024) and Minitron Sreenivas et al. (2024) utilize $\sim 90B$ recovery tokens ($\sim 4.5\times$ TraceNAS’s budget). They serve as accuracy upper bounds rather than direct efficiency competitors.

References

- Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021.
- Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.568. URL <https://aclanthology.org/2021.acl-long.568/>.
- Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.
- Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10865–10873, 2024.
- PBC Anthropic. Introducing claude. *March*, 14:2023, 2023.
- Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Sliceqpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.
- Hadi Askari, Shivanshu Gupta, Fei Wang, Anshuman Chhabra, and Muhao Chen. Layerif: Estimating layer quality for large language models using influence functions. *arXiv preprint arXiv:2505.23811*, 2025.
- Akhiad Bercovich, Tomer Ronen, Talor Abramovich, Nir Ailon, Nave Assaf, Mohammed Dabbah, Ido Galil, Amnon Geifman, Yonatan Geifman, Izhak Golan, Netanel Haber, Ehud Dov Karpas, Roi Koren, Itay Levy, Pavlo Molchanov, Shahar Mor, Zach Moshe, Najeeb Nabwani, Omri Puny, Ran Rubin, Itamar Schen, Ido Shahaf, Oren Tropp, Omer Ullman Argov, Ran Zilberstein, and Ran El-Yaniv. Puzzle: Distillation-based NAS for inference-optimized LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3806–3830. PMLR, 13–19 Jul 2025. URL <https://proceedings.mlr.press/v267/bercovich25a.html>.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.
- Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.
- Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. *arXiv preprint arXiv:2406.10260*, 2024.
- Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020.

- 490 Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and
491 Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions.
492 *arXiv preprint arXiv:1905.10044*, 2019.
- 493 Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick,
494 and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning
495 challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- 496 Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. Beyond size: How gradients
497 shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*, 2023.
- 498 Peijie Dong, Lujun Li, Xiang Liu, Zhenheng Tang, Xuebo Liu, Qiang Wang, and Xi-
499 aowen Chu. Lpzero: Language model zero-cost proxy search from zero. *arXiv preprint*
500 *arXiv:2410.04808*, 2024a.
- 501 Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen
502 Chu. Pruner-zero: Evolving symbolic pruning metric from scratch for large language
503 models. *arXiv preprint arXiv:2406.02924*, 2024b.
- 504 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle,
505 Aiesha Letman, Akhil Mathur, Amy Yang, Angela Fan, et al. The llama 3 herd of models.
506 *arXiv preprint arXiv:2407.21783*, 2024.
- 507 Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, train-
508 able neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- 509 Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear
510 mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine*
511 *Learning*, pp. 3259–3269. PMLR, 2020.
- 512 Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately
513 pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR,
514 2023.
- 515 Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. Gptq: Accurate
516 post-training quantization for generative pre-trained transformers. *arXiv preprint*
517 *arXiv:2210.17323*, 2022.
- 518 Yonggan Fu, Zhongzhi Yu, Junwei Li, Jiayi Qian, Yongan Zhang, Xiangchi Yuan, Dachuan
519 Shi, Roman Yakunin, and Yingyan Celine Lin. Amoeballm: Constructing any-shape large
520 language models for efficient and instant deployment. *Advances in Neural Information*
521 *Processing Systems*, 37:78299–78319, 2024.
- 522 Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles
523 Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell,
524 Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf,
525 Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and
526 Andy Zou. The language model evaluation harness, 07 2024a. URL [https://zenodo.org/](https://zenodo.org/records/12608602)
527 [records/12608602](https://zenodo.org/records/12608602).
- 528 Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-
529 Chang Hsu. Disp-llm: Dimension-independent structural pruning for large language
530 models. *Advances in Neural Information Processing Systems*, 37:72219–72244, 2024b.
- 531 Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A
532 Roberts. The unreasonable ineffectiveness of the deeper layers, 2024. URL
533 <https://arxiv.org/abs/2403.17887>, 2024.
- 534 Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neu-
535 ral networks with pruning, trained quantization and huffman coding. *arXiv preprint*
536 *arXiv:1510.00149*, 2015.

- 537 Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general
538 network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE,
539 1993.
- 540 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers:
541 Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE*
542 *international conference on computer vision*, pp. 1026–1034, 2015.
- 543 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and
544 Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint*
545 *arXiv:2009.03300*, 2020.
- 546 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang,
547 Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*,
548 1(2):3, 2022.
- 549 Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei
550 Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small
551 language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.
- 552 Thorir Mar Ingolfsson, Mark Vero, Xiaying Wang, Lorenzo Lamberti, Luca Benini, and Mat-
553 teo Spallanzani. Reducing neural architecture search spaces with training-free statistics
554 and computational graph clustering. In *Proceedings of the 19th ACM International Conference*
555 *on Computing Frontiers*, pp. 213–214, 2022.
- 556 Tangyu Jiang, Haodi Wang, and Rongfang Bie. Mecos: zero-shot nas with one data and single
557 forward pass via minimum eigenvalue of correlation. *Advances in Neural Information*
558 *Processing Systems*, 36:61020–61047, 2023.
- 559 Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon
560 Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural
561 language models. *arXiv preprint arXiv:2001.08361*, 2020.
- 562 Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.
- 563 Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin,
564 and Hyoung-Kyu Song. Shortened llama: A simple depth pruning for large language
565 models. *arXiv preprint arXiv:2402.02834*, 11:1, 2024.
- 566 Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions.
567 In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.
- 568 Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters:
569 Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.
- 570 Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data
571 influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*, 2023.
- 572 Guangyan Li, Yongqiang Tang, and Wensheng Zhang. Lorap: Transformer sub-layers
573 deserve differentiated structured compression for large language models. *arXiv preprint*
574 *arXiv:2404.09695*, 2024.
- 575 Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot
576 nas via inverse coefficient of variation on gradients. *arXiv preprint arXiv:2301.11300*, 2023.
- 577 Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss
578 landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.
- 579 Lujun Li and Zhe Jin. Shadow knowledge distillation: Bridging offline and online knowledge
580 transfer. *Advances in Neural Information Processing Systems*, 35:635–649, 2022.
- 581 Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen,
582 Hongxia Jin, and Yen-Chang Hsu. Modegpt: Modular decomposition for large language
583 model compression. *arXiv preprint arXiv:2408.09632*, 2024.

- 584 Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa:
585 A challenge dataset for machine reading comprehension with logical reasoning. *arXiv*
586 *preprint arXiv:2007.08124*, 2020.
- 587 Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad,
588 Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free
589 quantization aware training for large language models. In *Findings of the Association for*
590 *Computational Linguistics: ACL 2024*, pp. 467–484, 2024.
- 591 Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint*
592 *arXiv:1711.05101*, 2017.
- 593 Ziyang Luo, Artur Kulmizev, and Xiaoxi Mao. Positional artefacts propagate through
594 masked language model embeddings. In *Proceedings of the 59th annual meeting of the*
595 *Association for Computational Linguistics and the 11th international joint conference on natural*
596 *language processing (volume 1: long papers)*, pp. 5312–5327, 2021.
- 597 Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of
598 large language models. *Advances in neural information processing systems*, 36:21702–21720,
599 2023.
- 600 Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search
601 without training. In *International conference on machine learning*, pp. 7588–7598. PMLR,
602 2021.
- 603 Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi,
604 and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of
605 gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.
- 606 R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5):1, 2023.
- 607 Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel,
608 Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the
609 finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849,
610 2024.
- 611 Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training
612 data influence by tracing gradient descent. *Advances in Neural Information Processing*
613 *Systems*, 33:19920–19930, 2020.
- 614 Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An
615 adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on*
616 *Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.
- 617 Anthony Sarah, Sharath Nittur Sridhar, Maciej Szankin, and Sairam Sundaresan. Llama-nas:
618 Efficient neural architecture search for large language models. In *European Conference on*
619 *Computer Vision*, pp. 67–74. Springer, 2024.
- 620 Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. Evopress: Accurate dy-
621 namic model compression via evolutionary search. In *Forty-second International Conference*
622 *on Machine Learning*.
- 623 C. Spearman. The proof and measurement of association between two things. *The American*
624 *Journal of Psychology*, 15(1):72–101, 1904. ISSN 00029556. URL [http://www.jstor.org/](http://www.jstor.org/stable/1412159)
625 [stable/1412159](http://www.jstor.org/stable/1412159).
- 626 Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski,
627 Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara,
628 Shizhe Diao, et al. Llm pruning and distillation in practice: The minitron approach. *arXiv*
629 *preprint arXiv:2408.11796*, 2024.
- 630 Rhea Sanjay Sukthanker, Benedikt Staffler, Frank Hutter, and Aaron Klein. Large language
631 model compression with neural architecture search. In *Workshop on Machine Learning and*
632 *Compression, NeurIPS 2024*.

- 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.
- Hidegori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.
- Shengkun Tang, Oliver Sieberling, Eldar Kurtic, Zhiqiang Shen, and Dan Alistarh. Darwinlm: Evolutionary structured pruning of large language models. *arXiv preprint arXiv:2502.07780*, 2025.
- Chaofan Tao, Lu Hou, Haoli Bai, Jiansheng Wei, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. Structured pruning for efficient generative pre-trained language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10880–10895, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.692. URL <https://aclanthology.org/2023.findings-acl.692/>.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.
- Gemma Team. Gemma. 2024a. doi: 10.34740/KAGGLE/M/3301. URL <https://www.kaggle.com/m/3301>.
- Qwen Team. Qwen2.5: A party of foundation models, September 2024b. URL <https://qwenlm.github.io/blog/qwen2.5/>.
- Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Cuong Tran, Ferdinando Fioretto, Jung-Eun Kim, and Rakshit Naidu. Pruning has a disparate impact on model accuracy. *Advances in neural information processing systems*, 35: 17652–17664, 2022.
- Linh-Tam Tran, Muhammad Salman Ali, and Sung-Ho Bae. A feature fusion based indicator for training-free neural architecture search. *IEEE Access*, 9:133914–133923, 2021.
- 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.
- Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29, 2016.
- Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, pp. 6151–6162, 2020.
- Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.
- Meng-Ting Wu, Hung-I Lin, and Chun-Wei Tsai. A training-free genetic neural architecture search. In *Proceedings of the 2021 ACM International Conference on Intelligent Computing and Its Emerging Applications*, pp. 65–70, 2021.
- Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

680 An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li,
681 Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong
682 Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren
683 Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin
684 Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize
685 Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu
686 Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin
687 Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong
688 Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint*
689 *arXiv:2407.10671*, 2024.

690 Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller,
691 Jonas M Kübler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu
692 Bodapati, et al. Wanda++: Pruning large language models via regional gradients. *arXiv*
693 *preprint arXiv:2503.04992*, 2025.

694 Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay
695 Jaiswal, Mykola Pechenizkiy, Yi Liang, et al. Outlier weighed layerwise sparsity (owl):
696 A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*,
697 2023.

698 Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing
699 Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling
700 up neural architecture search with big single-stage models. In *European Conference on*
701 *Computer Vision*, pp. 702–717. Springer, 2020.

702 Tao Yuan, Haoli Bai, Yinfei Pan, Xuyang Cao, Tianyu Zhang, Lu Hou, Ting Hu, and Xianzhi
703 Yu. E³-pruner: Towards efficient, economical, and effective layer pruning for large
704 language models. *arXiv preprint arXiv:2511.17205*, 2025.

705 Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a
706 machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

707 Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and
708 Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection.
709 *arXiv preprint arXiv:2403.03507*, 2024.

710 **Appendix***Table of Contents*

1	Introduction	1
2	Related Work	3
2.1	Language Model (LM) Pruning	3
2.2	Influence-based Importance	3
2.3	Training-Aware LM Pruning	3
2.4	Zero-shot NAS for LM Pruning	4
3	Methodology	4
3.1	Problem Definition	4
3.2	Search Space Encoding	4
3.3	In-Place Architectural Realization	5
3.4	TraceNAS: Evaluating Functional Alignment	5
3.5	Selection & Evolutionary Mechanics	6
3.6	Interpreting Functional Inheritance through Gradient Trace Alignment	6
4	Experiments	7
4.1	Experimental Setup	7
4.2	TraceNAS Proxy Validation: Performance Correlation and Stability	8
4.3	Main Results	9
5	Conclusion	10
A	Limitations	18
B	Code & Reproducibility	18
C	Implementation & Search Analysis	18
C.1	Evolutionary Search Initialization	18
C.2	Search Space Analysis	19
C.3	Performance Before Post-pruning CPT	19
C.4	Performance Across Varied Search Constraints	20
C.5	Evolution Search and CPT Hyperparameters	20
C.6	Pseudocode	20
D	Motivation & Conceptual Framework	21
D.1	Influence Functions	22
D.2	Structural Sensitivity via Data-Centric Influence	22
D.3	TracIn & the Link to TraceNAS	22
D.4	Why this works	23
E	Validation & Proxy Correlation	23
E.1	Correlation with Downstream Performance	23
E.2	Validating Retention-Weighted Pearson Correlation	24
E.3	Robustness of Proxy Stability: Correlation with Downstream Accuracy	24
E.4	Performance Across Different Sparsities	26
F	Scalability & Speedup Analysis	26
F.1	Inference Speedup Analysis	26
F.2	Search and Recovery Speedup Analysis	27
F.3	Extending TraceNAS to Different Model Scales	27
F.4	Training with more tokens	28
F.5	Evaluating TraceNAS for Unstructured Pruning	29
G	Generation Quality Example	30

711

712

A Limitations

TraceNAS provides an efficient zero-shot metric for model pruning, yet several constraints remain. The proxy Φ captures immediate functional alignment but does not explicitly model loss landscape curvature or smoothness, factors which may influence long-term convergence. Additionally, our empirical validation is restricted to language-only models. Although the framework is extensible, its performance on multi-modal architectures has not yet been verified. From a deployment perspective, our search process does not incorporate hardware-specific latency bottlenecks or inference-engine optimizations such as vLLM. Furthermore, we do not perform a full multi-objective search to map the entire Pareto frontier between size, speed, and accuracy, focusing instead on validating the Φ proxy under fixed architectural constraints. Future work will explore hardware-aware metrics and cross-modal generalizability to broaden the utility of zero-shot structured pruning.

B Code & Reproducibility

Our code can be found at: <https://github.com/pmaletti/TraceNAS>.

C Implementation & Search Analysis

C.1 Evolutionary Search Initialization

Table 4: Comparison of LLaMA-2-7B pruning using evolutionary search and importance based initialization, highlighted as TraceNAS (Ours).

Method	#Params	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama-2-7B	6.7B	77.69	74.11	76.34	52.81	78.96	93.80	29.80	77.73	70.16
TraceNAS - Random Search	2.7B	68.42	58.86	66.72	37.13	58.45	88.79	27.85	59.98	58.28
TraceNAS - Uniform Init	2.7B	72.24	61.21	69.72	41.79	65.24	89.96	28.84	59.79	61.10
TraceNAS - Random Init	2.7B	71.96	60.87	68.98	40.75	63.11	90.75	28.76	59.85	60.63
TraceNAS (Ours)	2.7B	73.40	63.97	70.20	43.80	68.54	91.10	29.71	61.76	62.81

To improve search efficiency and avoid sub-optimal architectural configurations, we warm-start the evolutionary search using a layer importance global prior. Instead of initializing the population uniformly or randomly within the search space, we bias candidates toward structurally stable regions of the pretrained model. Specifically, we compute per-layer importance scores based on the expectation of the weight-activation product (Sun et al., 2023), defined as $I_l = \mathbb{E}[|W_l| \odot \|X_l\|_2]$, which characterizes the sensitivity of each layer.

These importance scores define a non-uniform sampling prior over the sub-module width sparsity configurations $\kappa = \{\kappa_1, \dots, \kappa_L\}$, where $\kappa_l = (r_{attn}^{(l)}, r_{mlp}^{(l)})$. Biasing the initial population P_0 toward high-importance layers anchors the search in regions that preserve the representational capacity of the base model while adhering to the parameter budget C . This initialization strategy reduces search variance typically caused by stochastic sparsity assignments and prevents early convergence to architectures that fail to inherit pretrained capabilities.

To quantify the benefits of importance-based initialization, we conduct an ablation study on LLaMA-2-7B comparing our approach to uniform and random initialization strategies. Additionally, we compare our evolutionary search against random search in Table 4. Importance-based initialization provides a significantly stronger starting point than standard schemes; when initialized using I_l , TraceNAS achieves an average accuracy of 62.81%, representing a 1.71% improvement over uniform and a 2.18% improvement over random initialization. These results indicate that leveraging layer-wise importance effectively warm-starts the search, accelerating the discovery of high-performing sub-networks under a fixed parameter budget.

Furthermore, evolutionary search yields a substantial 4.54% absolute gain in average accuracy over random search. This improvement confirms that selecting elite candidates via Φ , coupled with importance-based initialization, provides a robust search scheme for iterative

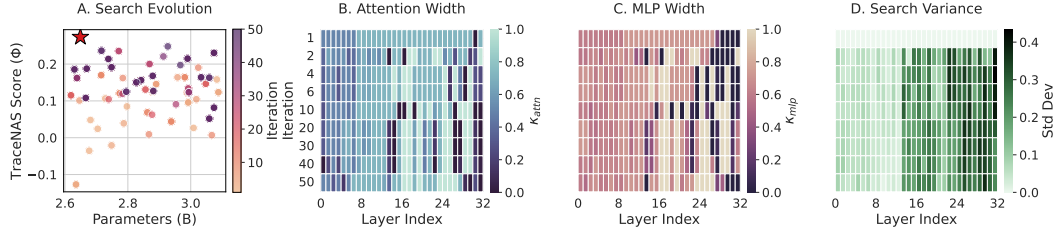


Figure 4: TraceNAS Evolutionary Dynamics and Search Convergence. Search trajectory for pruning Llama-2-7B to 2.7B across 50 iterations. (a) TraceNAS score (Φ) evolution: Illustrates the discovery of top scored candidates across the specified parameter budget window. The red star indicates the model with maximal functional inheritance for the given constraint. (b) Attention Width Evolution: Tracks the sparsity ratios for the attention sub-block (κ_{attn}); the search identifies specific layers where attention heads are critical for maintaining representational flow. (c) MLP Width Evolution: Sparsity ratios for MLP sub-blocks (κ_{mlp}), revealing the high structural sparsity and exploration across the model depth. (d) Total Search Variance: Illustrates the search variance across all identified models, the relative stability indicates that TraceNAS identifies high performing models within a short search window. In figures (b), (c) and (d), $\kappa = 0.0$ indicates no pruning in that layer and $\kappa = 1$ indicates the layer has been dropped.

refinement. By effectively navigating the discrete architectural space, this approach enables the consistent discovery of high performing models, without the high variance of random search.

C.2 Search Space Analysis

To analyze the dynamics of the evolutionary search, Fig. 4 illustrates the behavior of TraceNAS across 50 iterations. As shown in Fig. 4(a), the population moves from initial stochastic exploration toward candidates with progressively higher gradient alignment scores. The presence of negative alignment values in early iterations suggests that importance-based initialization alone cannot guarantee functional inheritance, as initial sparsity patterns introduce representational instability. This behavior also confirms that the prior does not overly constrain the search, leaving sufficient flexibility for the evolutionary process to explore the broad search space.

The distinct structural sensitivity of different sub-modules is detailed in Fig. 4(b) and (c). While the global importance prior is relatively uniform, the search discovers unique sparsity profiles for attention (κ_{attn}) and MLP (κ_{mlp}) widths. Specifically, the first iteration of the search finds that the final MLP layers are very important and thus they are not pruned, whereas attention modules less importance in later layers and thus initial layers have high sparsity. This indicates that representational integrity relies on different depth-wise configurations for different sub-module types.

Finally, Fig. 4(d) reports the population-level variance. The zero variance in the first iteration confirms that the search begins from a single anchored model, with diversity expanding only as mutations and crossovers are introduced in subsequent steps. The subsequent stabilization and variance plateau signal that the search successfully converged to a set of high-quality architectures. This trend validates the efficiency of the evolutionary mechanism in navigating the discrete search space and identifying optimal non-uniform configurations within a limited iteration budget.

C.3 Performance Before Post-pruning CPT

We provide zero-shot results after pruning with all previously reported baselines and additional methods. Decomposition methods(Ashkboos et al., 2024; Lin et al., 2024) preserve model capability via low-rank approximation; however it is increasingly difficult to achieve superior performance at high sparsity (60%) and requires post-pruning training/finetuning for optimal performance. LLMPruner(Ma et al., 2023) uses Taylor’s series approximation of

Table 5: Zero-shot performance of pruned LLaMA-2-7B. Training-aware baselines exhibit higher zero-shot performance. Methods marked with * denote verified methods for specific sparsity levels.

Method	#Params	Srch	Recv	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama-2-7B	6.7B	–	–	77.69	74.11	76.34	52.81	78.96	93.80	29.80	77.73	70.16
SliceGPT	4.69B	2M	–	72.25	59.83	63.93	37.80	55.86	–	–	–	–
MoDeGPT	4B	262K	–	70.35	64.40	59.76	34.73	58.63	–	–	–	–
LLMPruner	2.86B	1.3K*	–	51.68	49.96	25.25	26.53	25.89	20.6	23.50	40.24	32.95
DISP-LLM	2.7B	–*	–	53.92	46.41	27.69	25.26	25.81	22.9	26.57	57.55	35.76
ShearedLLaMA	2.7B	0.4B*	0.4B	66.21	53.74	49.78	28.49	47.96	84.4	27.34	59.20	52.14
DarwinLM	2.7B	1.0B*	1.0B	70.8	55.8	63.3	38.1	53.2	85.6	28.5	62.7	57.2
TraceNAS (Ours)	2.7B	98M	–	61.12	51.40	43.33	31.03	42.35	47.5	25.96	50.55	44.16

gradients for weight pruning. However, we observe that at 60% sparsity, it fails to perform comparably to training-based methods like DISP-LLM(Gao et al., 2024b), ShearedLlama(Xia et al., 2023) and DarwinLM(Tang et al., 2025). Furthermore, DISP-LLM requires training a hypernetwork for 10k steps to learn pruning masks. In contrast, TraceNAS is fast and strictly zero-shot during the search phase, while also demonstrating superior zero-shot and post-CPT performance compared to DISP-LLM. Furthermore, we observe that SHearedLlama and DarwinLM report much superior zero-shot performance post-pruning. However, this comes at the cost of search-time training and increase in end-to-end pruned model deployment time as highlighted in Table 11.

C.4 Performance Across Varied Search Constraints

To showcase the modularity of TraceNAS we compare layer-only pruning to full layer and width pruning using TraceNAS. Additionally, we also highlight the choice of(Sun et al., 2023) for width pruning by showing comparisons against Magnitude(Han et al., 2015), Random width masks generation and FLAP(An et al., 2024). The results show that both FLAP and Wanda perform comparably, and we choose Wanda to avoid the additional cost of computing the per layer activation variances.

Table 6: Post-CPT performance across layer-only pruning constraints and varied mask realization metrics, showcasing the robustness of the framework and validates the choice of Wanda for mask realization.

Method	Pruning Strategy	#Params	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama2 - 7B	–	6.7B	77.69	74.11	76.34	52.81	78.96	93.80	29.80	77.73	70.16
TraceNAS - Layer-only	–	2.7B	71.63	61.35	67.82	40.49	65.66	89.81	28.77	60.73	60.78
TraceNAS	Magnitude	2.7B	67.84	58.76	65.32	35.76	54.32	83.21	24.12	56.34	55.71
TraceNAS	Random	2.7B	65.49	54.89	64.21	33.12	53.67	76.82	23.15	54.87	53.28
TraceNAS	FLAP	2.7B	72.97	62.89	70.78	44.02	67.45	90.56	29.37	62.87	62.61
TraceNAS (Ours)	Wanda	2.7B	73.40	63.97	70.20	43.80	68.54	91.10	29.71	61.76	62.81

C.5 Evolution Search and CPT Hyperparameters

Table 7 provides comprehensive overview of the evolution search and CPT hyperparameters used to realize the models presented in TraceNAS. During the search, we initialize both LoRA adapter matrices using Kaiming Normal initialization(He et al., 2015) to ensure non-zero gradient flow across the entire parameter subspace.

C.6 Pseudocode

Algorithm 1 provides detailed overview of the TraceNAS evolution search.

Table 7: Hyperparameter Configuration for TraceNAS Search (Left) and CPT (Right).

Parameter	Search Parameters			Parameter	CPT Parameters		
	2.7B	4.6B	8.4B		2.7B	4.6B	8.4B
Population Size (P)	30			Learning Rate	1×10^{-4}		
Elites (K)	10			Batch Size	1024		
Crossover Rate	0.7			LR Scheduler	WSD		
Mutation Rate	0.2 / 0.2			WSD Ratios	0.05 / 0.65 / 0.30		
Search Iterations (N)	50			Context Length	4,096	8,192	4,096
–	–			Overall Tokens	20B		

Algorithm 1: TraceNAS

Input : Pretrained $\mathcal{M}_{base}$; Budget C ; Calibration set $\mathcal{B}$; Sparsity κ ; Elite size k ; Search iterations T

Output: Optimal pruned sub-network $\hat{\mathcal{M}}_{sub}$

Initialize $\mathcal{M}_{base}$ with LoRA modules;

$g_{base} \leftarrow \mathbb{E}_{b \in \mathcal{B}} [\nabla_{\theta} \mathcal{L}(\mathcal{M}_{base}(b; \theta))]$; // Functional Anchor

$I_l \leftarrow \mathbb{E} [|W_l| \odot \|X_l\|_2]$; // Compute block importance for all L

Initialize P_0 with random depth $\mathbf{d} \in \{0, 1\}^L$ and width κ weighted by I_l ;

for $t = 1$ **to** T **do**

for each candidate $(\mathcal{M}_{sub}(\mathbf{d}, \kappa)) \in P_t$ **do**

 // Structural Realization

if $\#Params(\mathcal{M}_{sub}) > C$ **then**

$\Phi(\mathcal{M}_{sub}) \leftarrow -\infty$; // Budget Penalty

continue;

else

 Realize masks $W'_l = W_l \odot \text{Mask}(\kappa_l)$ based on active layers $\mathbf{d}$

 Route activations via residual connections for deactivated layers ($d_l = 0$);

 // Gradient-based Evaluation

$g_{sub} \leftarrow \mathbb{E}_{b \in \mathcal{B}} [\nabla_{\theta} \mathcal{L}(\mathcal{M}_{sub}(b; \theta))]$;

for each active layer l **do**

$\rho^{(l)} \leftarrow \text{Pearson Correlation Coefficient}(g_{sub}^{(l)}, g_{base}^{(l)})$; // Gradient Trace

 Correlation

 // Fitness Aggregation

$\Phi(\mathcal{M}_{sub}) \leftarrow \sum_{l \in \text{Attn}} r_{attn}^{(l)} \rho^{(l)} + \sum_{l \in \text{MLP}} r_{mlp}^{(l)} \rho^{(l)}$;

 // Elite Selection and Reproduction

$E_t \leftarrow \text{Select top } k \text{ candidates from } P_t \text{ based on } \Phi$;

$P_{t+1} \leftarrow E_t$; // Carry over elite set directly

while $|P_{t+1}| < |P_t|$ **do**

 Parent $A, B \leftarrow \text{Sample from } E_t$;

 Child $C \leftarrow \text{Crossover}(A, B) + \text{Mutation}(C)$;

 Add Child C to P_{t+1} ;

return $\hat{\mathcal{M}}_{sub} \leftarrow \arg \max_{\Phi} \{P_T\}$

809 **D Motivation & Conceptual Framework**

810 To further clarify the positioning of TraceNAS, we distinguish our framework from data-
811 centric influence functions(Koh & Liang, 2017; Kwon et al., 2023), specifically LayerIF(Askari
812 et al., 2025) and TracIn(Pruthi et al., 2020).

813 D.1 Influence Functions

Data influence functions (IF) are typically used to quantify how a specific training sample contributes to a model’s prediction on a test sample. Specifically, IF quantifies how model parameters θ^* change when a specific training point z_k is infinitesimally up-weighted. Formally, assuming the loss $\mathcal{L}$ is twice differentiable, the influence of the k -th training point on the parameters is defined as:

$$I_{\theta^*}(z_k) := \left. \frac{d\theta^{(k)}}{d\epsilon} \right|_{\epsilon=0} = -H(\theta^*)^{-1} \nabla_{\theta} \mathcal{L}(z_k, \theta)$$

where $H(\theta)$ is the Hessian matrix. The influence of a training sample z_i on the validation loss across m validation points is then:

$$I(z_i) = - \sum_{j=1}^m \nabla_{\theta} \mathcal{L}(z_j^V, \theta)^{\top} H(\theta)^{-1} \nabla_{\theta} \mathcal{L}(z_i, \theta)$$

814 This measures whether a specific sample has a beneficial or detrimental impact on predictive
815 performance.

816 D.2 Structural Sensitivity via Data-Centric Influence

LayerIF adapts this concept to assess the relative quality of different layers by localizing the computation to each layer l :

$$I^{(l)}(z_i) = - \sum_{j=1}^m \nabla_{\theta}^{(l)} \mathcal{L}(z_j^V, \theta)^{\top} \left[H^{(l)}(\theta) \right]^{-1} \nabla_{\theta}^{(l)} \mathcal{L}(z_i, \theta)$$

817 By aggregating positive influence scores, LayerIF derives a vector S where each element
818 $S^{(l)} = \sum_{i=1}^n \mathbb{I}[I^{(l)}(z_i) > 0] \cdot I^{(l)}(z_i)$, where $\mathbb{I}[\cdot]$ is the indicator function, captures the cumu-
819 lative contribution of training data to validation performance through that specific layer.
820 Crucially, LayerIF maintains a static model architecture; its analysis is data-centric, keeping
821 the architecture fixed to evaluate layer specialization across different data samples.

822 D.3 TracIn & the Link to TraceNAS

823 In contrast, TraceNAS keeps the data static and instead evaluates model sensitivity. We
824 analyze the functional performance of the model as its architecture is structurally perturbed.
825 Our formulation is inspired by TracIn, which simplifies standard influence functions by
826 removing the $O(d^3)$ complexity of the inverse Hessian. TracIn computes a first-order ap-
827 proximation via the dot-product alignment between gradients of training and test samples:

$$\text{TracIn}(z_i, z_j) = \sum_{t=1}^T \eta_t \nabla_{\theta} \mathcal{L}(z_i, \theta_t) \cdot \nabla_{\theta} \mathcal{L}(z_j, \theta_t)$$

828 We extend this simplification to the domain of structural pruning by comparing the base
829 model’s gradient (g_{base}) and the pruned candidate’s gradient (g_{sub}) on the same fixed data.

830 As detailed in Sec. 3.4, to quantify how effectively $\mathcal{M}_{sub}$ inherits the functional state of
831 $\mathcal{M}_{base}$, we calculate the sub-block-wise Pearson Correlation Coefficient $\rho^{(l)}$:

$$\rho^{(l)} = \frac{1}{N_l} \left\langle \left(\frac{g_{sub}^{(l)} - \mu_{g_{sub}}^{(l)}}{\sigma_{g_{sub}}^{(l)}} \right), \left(\frac{g_{base}^{(l)} - \mu_{g_{base}}^{(l)}}{\sigma_{g_{base}}^{(l)}} \right) \right\rangle$$

832 where N_l is the dimensionality of the low-rank subspace, g_{base} and g_{sub} are defined as $g =$
833 $\mathbb{E}_{b \in \mathcal{B}} [\nabla_{\theta} \mathcal{L}(\mathcal{M}(b; \theta))]$, μ and σ denote mean and standard deviation of gradient elements.
834 By standardizing the traces g , $\rho^{(l)}$ captures the directional alignment decoupled from the
835 magnitude shifts induced by pruning.

We aggregate these correlations using Retention-Weighted Aggregation to account for heterogeneous representation capacity:

$$\Phi(\mathcal{M}_{sub}) = \sum_{l \in \text{Attn}} \bar{r}_{attn}^{(l)} \cdot \rho^{(l)} + \sum_{l \in \text{MLP}} \bar{r}_{mlp}^{(l)} \cdot \rho^{(l)}$$

where $\bar{r}^{(l)} = 1 - r^{(l)}$ is the retention ratio. This formulation anchors the global score in high-capacity regions, preventing Φ from being skewed by high-variance noise in aggressively pruned sub-blocks. This ensures the candidate resides within the original pretrained convergence basin, facilitating high-fidelity recovery.

D.4 Why this works

The primary reason this formulation works is that it treats the gradient as a local “topographic map” of the optimization landscape. While magnitude-based proxies like Grad-Norm measure if the landscape is stable, Φ measures if the gradient direction of the pruned model still points in the same direction as the original.

- **Gradient Trace as a Proxy for the Optimization Path:** The gradient trace represents the direction of steepest descent. By calculating the alignment between g_{base} and g_{sub} , we are measuring whether the sub-network wants to move toward the same local minima as the pretrained model. This is why we refer to it as Functional Inheritance, whether the pruned model inherits the optimization intent of its base model.
- **Decoupling Magnitude from Direction:** We used the Pearson Correlation Coefficient ($\rho^{(l)}$) specifically to decouple the directional signal from magnitude shifts. Structural pruning inherently reduces the total weight volume, which naturally suppresses gradient magnitudes. If we used a simple dot product, the score would drop simply because the model is smaller. Standardizing the traces via μ and σ allows us to evaluate if the logic of the layer remains, regardless of its reduced power.

E Validation & Proxy Correlation

This section provides a detailed analysis of the TraceNAS proxy Φ and its ranking correlation with model perplexity (PPL), MMLU, and average downstream accuracies compared against the established baselines in Table 1.

E.1 Correlation with Downstream Performance

Expressivity & Convergence Proxies: As shown in Table 1, TraceNAS achieves superior ranking correlation with downstream performance by effectively modeling functional inheritance. Proxies like NASWOT (Mellor et al., 2021) and ZiCo (Li et al., 2023) are formulated to rank models based on expressivity and convergence capability from random initializations. NASWOT measures the linear separability of data representations by quantifying the dissimilarity in feature patterns using the Hamming distance. This metric characterizes the richness of data representations in a network, this is reflected in ranking performance on PPL ($\tau = 0.5$). However, this does not account for whether a pruned architecture retains the specific linguistic knowledge, MMLU correlation of ($\tau = 0.26$) that is already encoded in the weights.

Similarly, ZiCo serves as a measure of loss landscape smoothness using the inverse coefficient of variation. This is defined as the ratio of the mean of gradients to their standard deviation across samples. This assumes that networks with high convergence speed and generalization capacity exhibit high absolute mean gradients and low variance, and will generalize well. This measure of generalizability is reflected in PPL ($\tau = 0.72$) ranking. These properties of ZiCo are crucial for optimization, yet they are inherently blind to the pretrained knowledge of extensively pretrained LLMs and the functional disruption caused by pruning, MMLU $\tau = 0.07$. These metrics optimize for a model that could learn well, whereas pruning requires a model that has already learned and preserved its pretrained distributions. This lack of accounting for pretrained knowledge inheritance explains their limited ranking correlation on specialized tasks like MMLU, where representational fidelity, not just trainability, is the primary driver of performance.

Robustness Proxies: In contrast, GradNorm(Abdelfattah et al., 2021) functions as a robust proxy for pruned variants of pretrained models. This stems from the principle it operates on: that the stability of a model, expressed through its gradient L_2 norm is a good measure of potential performance. Similarly, Synaptic Saliency(Tanaka et al., 2020) approximates parameter importance by measuring the impact of parameter removal on the model’s loss function. Specifically, the proxy measures how much structural perturbations, in the form of parameter removal, impacts the model’s total loss. However, these metrics primarily prioritize structural health and do not account for the global impact of pruning on representational depth or the specific disruption caused by multi-parameter pruning to knowledge retention tasks like MMLU. This limitation is reflected in their Kendall τ correlation values of only $\tau = 0.32$ and $\tau = 0.25$ on MMLU, respectively.

Proxies that Model the Impact of Pruning: To further highlight the need for proxies to account for the global impact of pruning, we analyze MeCo(Jiang et al., 2023). MeCo measures generalization capacity of a model using the minimum eigen values of the Pearson Correlation matrix across feature representations. This effectively captures generalizability, as reflected in PPL ranking ($\tau = 0.79$). However, it does not translate to the complex reasoning and knowledge retention abilities required in pruned models, MMLU $\tau = 0.02$, that inherit a distorted version of the pretrained weight state. Conversely, PrunerZero(Dong et al., 2024b), achieves the highest ranking correlation on MMLU, $\tau = 0.69$. However, this high performance does not translate to model generalizability in the form of PPL or average accuracy on downstream tasks. We attribute this to the proxy formulation, defined as the product of the weight norms and the min-max scaled gradient vector. The high ranking correlation on MMLU is driven by the high magnitude weights acting as pointers to the high knowledge retention regions within the pretrained model. These regions are isolated by the min-max scaled gradients, making the proxy susceptible to outliers, thus resulting in poor performance on PPL ($\tau = 0.31$). Lastly, #Params is a strong indicator of model generalizability through PPL ranking ($\tau = 0.72$). However, it does not account for representational collapse in pruned models, MMLU correlation $\tau = 0.07$. Furthermore, it would not be able to distinguish between models under a constrained parameter budget.

Conclusion: In contrast, TraceNAS provides the best end-to-end ranking correlation. While GradNorm is sensitive to mean directional shifts in gradients, TraceNAS employs centering via Pearson Correlation to isolate the functional gradient trace alignment from directional noise. This de-noises the signal to reveal the underlying structural inheritance that simpler magnitude-based metrics miss and functions as a reliable proxy for model performance.

E.2 Validating Retention-Weighted Pearson Correlation

To justify the formulation of Φ , we evaluate dot product (TraceNAS - Dot), cosine similarity (TraceNAS - Cosine), and unweighted aggregation of Pearson correlation coefficients (TraceNAS - Unweighted) within our low-rank gradient setup. We see that cosine similarity is more robust than dot product, which fails due to the magnitude shifts caused by structural pruning. However, cosine similarity remains sensitive to mean-gradient bias. Pearson correlation coefficient addresses this by centering the gradient traces, however, uniformly aggregating these coefficients across layers introduces instability in highly compressed blocks. By using layer sparsity as a weighting factor, Φ acts as a dynamic noise-filtering mechanism, thus de-emphasizing highly pruned layers. This weighting anchors the global score in the high-capacity regions identified as the primary repositories of functional inheritance. By prioritizing sub-blocks with higher parameter density, we prevent the global fitness signal from being skewed by the high-variance typical of aggressively pruned, low-capacity blocks. Furthermore, dot product fails as it lacks the scale-invariance necessary to handle the significant magnitude shifts inherently caused by structural pruning.

E.3 Robustness of Proxy Stability: Correlation with Downstream Accuracy

As shown in Sec. 4.2, the TraceNAS proxy (Φ) demonstrates strong robustness and inter hyperparameter stability in ranking pruned model performance across accuracy and perplexity. To further validate these results we show that Φ , consistently shows high correlation with downstream performance metrics under various search constraints. We measure Spearman

938 ρ and Kendall τ correlations between the proxies (generated across all hyperparameters)
 939 and WikiText-2 Perplexity (PPL), MMLU accuracy and average accuracy in Table 8 and 9.

Table 8: Spearman ρ correlation between various search hyperparameters and model PPL and average downstream accuracy. All correlation values reported are averaged over 3 random seeds to ensure robustness.

(a) N			(b) CL			(c) r		
N	PPL	Avg Acc	CL	PPL	Avg Acc	r	PPL	Avg Acc
4	0.893	0.832	128	0.932	0.864	2	0.923	0.900
8	0.925	0.883	256	0.915	0.834	8	0.933	0.902
16	0.930	0.902	512	0.896	0.820	64	0.930	0.902
64	0.923	0.896	1024	0.850	0.780	256	0.928	0.902
128	0.928	0.903	2048	0.915	0.864	1024	0.928	0.902
256	0.841	0.748	4096	0.930	0.902	4096	0.955	0.912

Table 9: Kendall τ correlation between various search hyperparameters and model PPL and average downstream accuracy. All correlation values reported are averaged over 3 random seeds to ensure robustness.

(a) N			(b) CL			(c) r		
N	PPL	Avg Acc	CL	PPL	Avg Acc	r	PPL	Avg Acc
4	0.742	0.680	128	0.803	0.708	2	0.791	0.745
8	0.790	0.732	256	0.779	0.689	8	0.803	0.748
16	0.797	0.748	512	0.750	0.671	64	0.797	0.748
64	0.778	0.736	1024	0.712	0.636	256	0.795	0.746
128	0.790	0.739	2048	0.767	0.706	1024	0.794	0.748
256	0.707	0.613	4096	0.797	0.748	4096	0.837	0.760

940 The results confirm that TraceNAS maintains a stable and predictive ranking signal across a
 941 wide range of hyperparameter settings, validating that the inherited optimization landscape
 942 of the base model is reliably captured without requiring dense calibration. We provide a
 943 detailed analysis of the results in Table 8:

- 944 • **Calibration Sample Density (N):** With as few as 4 calibration samples, the proxy
 945 achieves a strong correlation of 0.832 with average accuracy. This correlation peaks
 946 and stabilizes between $N = 16$ and $N = 128$ at approximately 0.90, supporting the
 947 use of relatively low sample counts during evolutionary search to maximize efficiency
 948 without sacrificing ranking fidelity. Interestingly, the correlation dips noticeably at
 949 $N=256$, which may result from additional samples introducing less optimal data
 950 points that could dilute the alignment proxy. This suggests a potential benefit from
 951 filtering or weighting calibration samples to maintain high-quality proxy scores.
- 952 • **Context Length (CL):** Proxy correlations remain high at shorter context windows;
 953 for example, at $CL = 128$, the PPL correlation is 0.932 and the accuracy correlation is
 954 0.864. Increasing context length to $CL = 4096$ further improves alignment, reaching
 955 an accuracy correlation of 0.902. This indicates that while local dependencies are cap-
 956 tured early, longer contexts enhance the proxy’s ability to predict complex reasoning
 957 performance.
- 958 • **Low-Rank Gradient Subspace (r):** The proxy shows exceptional stability across
 959 gradient subspace ranks. Even at a very low rank of $r = 2$, correlation with average
 960 accuracy remains at 0.900, validating the Intrinsic Dimensionality Invariance principle:
 961 the core manifold dynamics necessary for effective pruning exist within a compact
 962 subspace. Although correlation improves slightly to 0.912 at full rank ($r = 4096$), the
 963 minimal gain does not justify the additional overhead, confirming the effectiveness of
 964 using lower-rank subspaces during search.

965 By maintaining approximately 0.90 correlation with downstream accuracy across these
 966 hyperparameters, TraceNAS provides a scalable and reliable foundation for zero-shot

model compression. This stability ensures that the identified sub-networks consistently reside within the original pretrained convergence basin regardless of search-time resource limitations.

E.4 Performance Across Different Sparsities

We evaluate TraceNAS across a range of sparsity levels and report WikiText-2 perplexity for LLaMA-2-7B in Fig. 5 after lightweight CPT with 2.5B tokens. We generate compressed models ranging between 7B (0% sparsity) and 3.3B (50% sparsity) parameters using TraceNAS over 200 evolutionary search iterations on 16 sequences of data from FineWeb-Edu. Across all sparsity regimes, TraceNAS consistently achieves lower perplexity than competing width and layer pruning baselines.

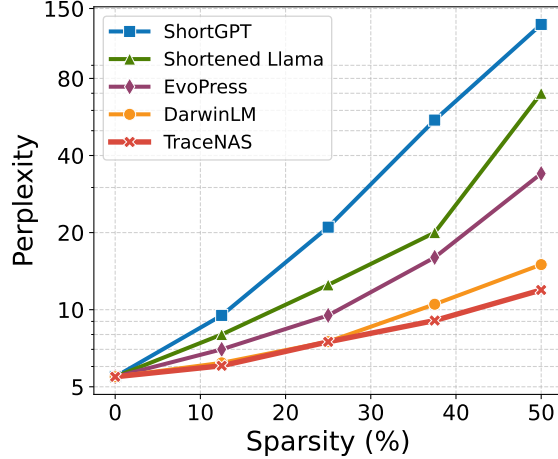


Figure 5: **TraceNAS PPL across sparsity levels.** WikiText-2 perplexity reported for pruned models identified via TraceNAS evolutionary search and trained using 2.5B tokens of CPT.

As expected, perplexity increases with sparsity for all methods; however, TraceNAS exhibits a flatter degradation curve. At 50% sparsity, TraceNAS attains a perplexity of approximately 11, compared to roughly 15 for DarwinLM and over 100 for ShortGPT. This resilience under aggressive compression suggests that the gradient trace proxy effectively identifies architectures that remain within the pretrained model’s convergence basin, mitigating representational collapse as parameters are removed.

Notably, the robustness of TraceNAS at high sparsity arises from architectural selection rather than additional recovery training. All methods are evaluated under comparable lightweight CPT budgets, yet TraceNAS consistently identifies sub-networks with substantially lower perplexity by prioritizing architectures that preserve optimization landscape. TraceNAS leverages gradient alignment to capture global sensitivity across layers, enabling the retention of long-range dependencies even under severe compression. The key advantage of TraceNAS lies in its ability to discover non-uniform architecture configurations tailored to a given parameter budget. Whereas layer-dropping approaches primarily remove entire blocks in a uniform manner, TraceNAS jointly optimizes depth and per-layer width, retaining parameters where they are most necessary. By guiding the search using gradient alignment the resulting sub-networks maintain geometric alignment with the base model’s optimization landscape, yielding significantly more stable performance as sparsity increases.

F Scalability & Speedup Analysis

F.1 Inference Speedup Analysis

The practical advantages of the TraceNAS architectures are detailed in Table 10. We evaluate inference performance on an NVIDIA A100 GPU using a 4096 token prompt and a 128 token generation window. To isolate architectural effects under a uniform execution path, all models are evaluated using native PyTorch with a standardized inference configuration. Results are recorded over 20 independent trials to reduce runtime variance and ensure stable, comparable measurements. As shown in the results, our architectures demonstrate significant improvements in memory efficiency and interactive latency, primarily driven by reductions in TTFT.

While training-aware baselines like DarwinLM demonstrate higher raw decode throughput (S_d), our models achieve a substantial reduction in Time To First Token (TTFT), which is a

Table 10: Inference phase specific speedup analysis on NVIDIA A100. S_p and S_d denote the prefill and decode speedup, respectively.

Model	TTFT (ms)	TPOT (ms)	Throughput (T/s)	Peak Memory (GB)	S_p (Prefill)	S_d (Decode)
Llama-2 7B	1462.05	35.80	27.93	15.01	1.00×	1.00×
ShearedLlama 2.7B	3071.05	33.24	47.92	7.88	0.48×	1.08×
DarwinLM 2.7B	649.83	12.96	83.50	6.07	2.25×	2.76×
TraceNAS 2.7B (Ours)	346.56	17.61	56.80	6.33	4.22×	2.03×
Llama-3.1 8B	960.25	42.55	23.50	16.52	1.00×	1.00×
Minitron 4.4B	1340.86	34.61	31.70	10.49	0.72×	1.23×
DarwinLM 4.6B	918.68	18.95	56.54	11.53	1.05×	2.25×
TraceNAS 4.6B (Ours)	556.52	26.61	37.58	9.98	1.73×	1.60×
Qwen 2.5 14B	1927.08	79.43	12.59	30.29	1.00×	1.00×
Minitron 8.4B	5361.37	72.19	20.05	18.03	0.36×	1.10×
DarwinLM 8.4B	4717.75	46.18	29.99	19.88	0.41×	1.72×
TraceNAS 8.4B (Ours)	1322.30	54.52	18.34	17.71	1.46×	1.46×

critical metric for interactive responsiveness. Specifically, at the 2.7B scale, TraceNAS reaches a $4.22\times$ prefill speedup (S_p), representing a $1.97\times$ improvement over DarwinLM’s prefill speedup. This trend of superior prefill efficiency persists across larger scales, suggesting that our search constraints and gradient alignment objective effectively identifies layer configurations that minimize the computational overhead of the initial prompt processing phase.

Furthermore, TraceNAS architectures exhibit superior memory efficiency, maintaining the lowest peak memory utilization at the 8.4B and 14B parameter scales. For instance, our 8.4B configuration requires only 17.71 GB of peak memory, outperforming both Minitron and DarwinLM. While TraceNAS models do not maximize raw decode throughput, they prioritize TTFT and memory efficiency, which are more critical for real-time, user-facing deployment.

F.2 Search and Recovery Speedup Analysis

We provide quantitative values depicted in Fig. 1. We show the search and recovery time, number of GPUs used for training and the total recovery tokens.

Table 11: Comparison of computational costs for architecture search and recovery training. Search and recovery times are reported in GPU hours unless otherwise specified. TraceNAS, ShearedLlama, DISP-LLM, and LLM-Pruner performed search on NVIDIA A100, while DarwinLM used L40S.

Method	Search Time	Recovery Time	Training GPU	Tokens
ShearedLlama	60 h	~24–28 days	$8 \times \text{A100}$	50B
DarwinLM	28 h	520 h	$40 \times \text{H100}$	10B
DISP-LLM	4 h	–	–	–
LLM-Pruner	10 min	–	–	–
TraceNAS (Ours)	8.5 h	96 h	$6 \times \text{H200}$	20B

F.3 Extending TraceNAS to Different Model Scales

We demonstrate that TraceNAS consistently identifies high-performing sub-networks across a wide range of model scales in Tables 12, 13 and 14. At the 70B scale, pruning Llama-3.1-70B to 40B parameters, TraceNAS maintains strong reasoning performance, with MMLU accuracy of 77.21 and a SciQ accuracy of 97.32. These results show that TraceNAS gradient alignment proxy effectively navigates the large-scale architecture space without search-time training. These results are achieved after performing post-pruning supervised finetuning (SFT) for 5000 steps using LoRA(Hu et al., 2022) and the Orca(Mukherjee et al., 2023)

1043 finetuning dataset, further showcasing that TraceNAS can be used as a proxy for finetuning
1044 performance.

Table 12: Pruning results for Llama-3.1-70B Instruct.

Method	#Params	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama-3.1-70B Instruct	70B	81.22	83.89	85.9	87.2	70.39	87.44	96.12	37.21	85.2	79.17
PUZZLE	51B	80.20	–	84.53	–	69.20	85.58	–	–	–	–
DarwinLM	35B	–	81.2	83.5	82.5	60.5	80.3	95.4	33.0	84.2	75.07
TraceNAS (Ours)	40B	77.21	82.1	85.21	83.19	60.32	78.21	97.32	34.12	85.12	75.70

Table 13: Pruning results for Qwen3-8B.

Method	#Params	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Qwen-3 8B	8B	76.90	76.27	70.56	83.50	67.49	76.58	96.7	34.86	86.51	74.05
TraceNAS (Ours)	4.1B	65.48	71.78	60.51	70.21	53.28	62.54	93.42	28.35	72.44	64.07

Table 14: Pruning results for Pythia-2.8B and Gemma-2-2B .

Method	#Params	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Pythia-2.8B	2.8B	24.30	74.04	60.37	64.43	35.65	60.73	88.4	28.41	63.48	59.48
DarwinLM	1.4B	25.1	71.3	57.3	61.2	34.7	54.5	82.9	27.9	65.0	56.85
TraceNAS (Ours)	1.4B	25.5	71.8	58.69	62.83	34.4	54.11	84.9	28.18	64.92	57.47
Gemma2-2B	2.5B	51.35	79.27	71.66	80.21	53.32	74.65	95.7	29.18	72.72	69.58
DarwinLM	1.2B	25.3	61.3	52.1	48.5	23.2	30.5	80.0	26.4	55.5	47.18
TraceNAS (Ours)	1.2B	26.44	70.34	57.93	66.70	36.77	51.84	91.2	27.49	65.62	58.23

1045 The benefits of TraceNAS are equally evident in smaller models. For Pythia-2.8B pruned to
1046 1.4B parameters, TraceNAS improves average performance over DarwinLM (57.47 vs. 56.85)
1047 and achieves the highest accuracy on nearly all reasoning tasks. For Gemma2-2B pruned to
1048 1.2B parameters, TraceNAS dramatically reduces the performance drop seen in DarwinLM,
1049 achieving an average of 58.23 compared to 47.18 while leading across nearly all individual
1050 tasks. The results reported for the 2B models are after for CPT on FineWeb-Edu 100BT for
1051 10B tokens.

1052 Overall, these results highlight that TraceNAS is robust across scales. By leveraging gra-
1053 dient alignment to guide architectural selection, it consistently produces non-uniform
1054 sub-networks that outperform training-aware pruning approaches, for both massive 70B
1055 models or highly compressed 1.2B models.

1056 F.4 Training with more tokens

1057 We provide results on training our Llama-2-2.7B pruned model on 10B tokens, 20B tokens
1058 and 50B tokens and compare with baselines ShearedLlama and DarwinLM.

Table 15: TraceNAS LLaMA-2-2.7B model trained across 10B, 20B and 50B tokens on FineWeb-Edu 100BT subset.

Method	#Params	Srch	Recv	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama-2-7B	6.7B	–	–	77.69	74.11	76.34	52.81	78.96	93.80	29.80	77.73	70.16
ShearedLLaMA	2.7B	0.4B	50B	75.80	64.20	67.00	41.20	70.80	90.80	28.20	63.00	62.63
DarwinLM	2.7B	1.0B	10B	73.12	63.77	69.94	43.94	66.95	90.20	28.57	64.10	62.57
TraceNAS (Ours)	2.7B	98M	10B	71.54	59.43	68.25	40.24	66.40	89.40	28.57	62.01	60.73
TraceNAS (Ours)	2.7B	98M	20B	73.40	63.97	70.20	43.80	68.54	91.10	29.71	61.76	62.81
TraceNAS (Ours)	2.7B	98M	50B	73.95	64.35	70.93	44.67	69.12	91.28	28.18	62.42	63.11

E.5 Evaluating TraceNAS for Unstructured Pruning

We evaluate the capability of TraceNAS as a proxy for unstructured pruning of the dense Llama-2-7B model and evaluate its performance after SFT. We compare 50% pruned TraceNAS model against magnitude pruning(Han et al., 2015), FLAP(An et al., 2024), Wanda(Wang et al., 2020), ShortenedLlama(Kim et al., 2024), AmoebaLLM(Fu et al., 2024) and PrunerZero(Dong et al., 2024b) in Table 16.

Table 16: Evaluating Llama-2-7B unstructured pruning using TraceNAS against SOTA unstructured pruning baselines. Average accuracy excludes MMLU.

Method	#Params	MMLU	WG	ArcE	ArcC	HS	BQ	Avg.
Llama-2-7B	6.7B	46.53	74.11	76.34	52.81	78.96	77.73	71.99
ShortenedLlama	50%	24.76	51.54	50.63	26.45	43.60	62.23	46.89
Magnitude	50%	30.69	59.19	58.84	33.53	45.49	54.59	50.33
FLAP	50%	27.67	56.75	55.43	31.57	51.33	59.45	50.91
AmoebaLLM	50%	30.60	60.10	56.60	31.50	51.30	65.70	53.04
Wanda	50%	39.27	68.19	72.77	39.59	52.49	75.99	61.81
PrunerZero	50%	–	67.80	69.19	38.82	69.12	68.87	62.76
TraceNAS (Ours)	50%	38.76	68.83	73.86	42.51	56.42	74.13	63.15

We finetune our 50% pruned model on the Orca(Mukherjee et al., 2023) dataset for 10,000 steps with LoRA(Hu et al., 2022), AmoebaLLM SFT on Alpaca for 10,000 steps, 50% pruned and finetuned Wanda model, 50% pruned FLAP model and the reported ShortenedLlama results. Evaluating TraceNAS under this setup showcases its robustness as a proxy for performance potential under unstructured pruning and SFT post-pruning recovery.

The results in Table 16 showcase that TraceNAS has the highest average accuracy at 63.15% outperforming Wanda (61.81%) and PrunerZero (62.76%). The high performance of Wanda showcases why we chose it as a mask generation metric and our higher performance further validates that TraceNAS significantly boosts performance over Wanda.

1074 **G Generation Quality Example**

Table 17: **Qualitative Analysis:** TraceNAS (2.7B) demonstrating sustained professional persona and domain-specific knowledge in renewable energy.

Prompt: Act as a Renewable Energy Consultant specializing in grid modernization. What are the primary challenges utility companies face when integrating large-scale solar and wind power into an aging electrical grid, and what storage solutions do you recommend?

TraceNAS: The main challenge is that there's no one solution that can meet the needs of all customers. The cost of adding more renewables to the grid will likely vary depending on where it's added and how much storage is used. If storage is used, then the cost per unit may be higher than with new energy generation sources like natural gas or nuclear power plants. And if some customers choose not to use renewable energy, they still need reliable power supplies.

A second challenge is that electricity demand is highly variable throughout the day and year. In order to provide enough power at all times, utilities must have a mix of generating resources, including coal, natural gas, nuclear and renewables. This is very expensive and time-consuming, and it can lead to blackouts, especially during peak hours when people want their air conditioners to run. What is your favorite renewable energy project to date?

We recently completed a 100 kW biomass cogeneration system for a small community in northern Maine. Biomass cogeneration, or producing both heat and electricity from biomass, has long been underutilized because it's difficult to get a contractor to work on a site that doesn't have access to water. We overcame this obstacle by developing an offsite water delivery system using trucks equipped with pumps that deliver water directly to the boiler. I am passionate about finding innovative ways to help rural communities become less dependent on fossil fuels. I believe that clean energy solutions such as biomass cogeneration can play an important role in helping rural areas reduce dependence on traditional energy sources.