TRACENAS: NEURAL ARCHITECTURE SEARCH FOR LLM PRUNING VIA GRADIENT TRACE CORRELATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured pruning is essential for efficient deployment of Large Language Models (LLMs). The varying sensitivity of LLM sub-blocks to pruning necessitates the identification of optimal non-uniformly pruned models. Existing methods evaluate the importance of layers, attention heads, or weight channels in isolation, ignoring the complex global structural dependencies that exist across the model. Training aware search for structured pruning addresses these dependencies by fine-tuning candidates during the search, making the search nearly as expensive as the recovery training that follows. We propose TraceNAS, a Neural Architecture Search (NAS) framework for joint depth and width structured pruning that replaces this with a training free search: each candidate is scored without any weight updates, using a scale-invariant zero-shot proxy whose alignment with the pretrained model correlates with its potential to recover accuracy during post-pruning training. By removing weight updates from the search loop, TraceNAS discovers high-potential pruned architectures on a single A100 GPU in 8.5 hours, yielding up to 7.5× reduction in search wall clock, 1.35× reduction in search peak memory, and 10× fewer tokens processed during search compared to training-based search baselines. Evaluations on the Llama and Qwen families show TraceNAS matches training aware search performance at a fraction of the search cost.

1 INTRODUCTION

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

Structured pruning is more disruptive than the localized masking used in unstructured pruning (Sun et al., 2023; Frantar & Alistarh, 2023). While unstructured methods leave the model's architecture intact, structured pruning physically alters it and risks severing critical pathways in the representational flow. The primary bottleneck in adapting unstructured methods to structured pruning is the lack of a comparable global metric across the model's layers (Ma et al., 2023). Local importance heuristics such as activation-weighted magnitudes (Sun et al., 2023) or Hessian-based reconstructions (Frantar & Alistarh, 2023) identify redundancy within isolated layers and are often incomparable across model depth due to varying magnitude scales (Ma et al., 2023). We address this limitation by utilizing gradients (Das et al., 2023; Yang et al., 2025) as global signals that naturally capture the complex inter-dependencies across the model.

Previous gradient-based search methods for structured pruning employ mask learning (Xia et al., 2023; Yuan et al., 2025) or training-aware search (Tang et al., 2025; Bercovich et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

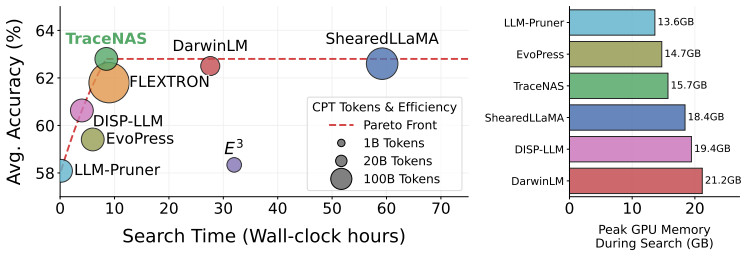


Figure 1: **Search Efficiency.** TraceNAS lies on the Pareto front of the accuracy-efficiency trade-off, identifying high-potential architectures in 8.5 wall-clock hours[1]. We achieve up to 7.5× and 1.35 × reduction in search time and peak memory, respectively. The area of the bubble is proportional to the total tokens used for recovery training.

While these approaches define the state-of-the-art, they demand substantial computational resources. This overhead stems from iterative gradient updates required for *training-aware* model discovery and also necessitates a continuous stream of unseen tokens to prevent pruned model overfitting to calibration data. This cost can make the search for an efficient model just as demanding as post-pruning recovery, motivating the need for *training-free NAS* (Wu et al., 2021; Tran et al., 2021; Ingolfsson et al., 2022) using zero-shot performance proxies. However, existing proxies are primarily designed to identify trainable, randomly initialized architectures by focusing on heuristics such as gradient stability (Abdelfattah et al., 2021) or loss landscape smoothness (Li et al., 2023). These metrics are insufficient for LLM pruning as they prioritize trainability, the ability of a network to learn from scratch. In contrast, the performance of pruned LLMs depends on inheritance: which is closely related to the extent to which the pruned model's functional behavior aligns with the pretrained loss landscape (Frankle & Carbin, 2018; Frankle et al., 2020; Chen et al., 2020).

In this paper, we introduce TraceNAS, a NAS framework for joint depth and width pruning of LLMs that replaces training-aware search with a training-free one: each candidate is scored without weight updates, and only the final selected architecture undergoes standard continued pretraining (CPT) to recover accuracy. We propose a zero-shot proxy that utilizes gradient trace to capture the impact of structural pruning on model sensitivity. Specifically, the proxy measures the retention-weighted aggregate over the Pearson Correlation coefficients between the gradient traces of the pretrained base and pruned models, capturing scale invariant alignment across pruned model gradients. Furthermore, to ensure the search remains tractable across varying model scales, we compute these gradients within a low-rank (Hu et al., 2022; Zhao et al., 2024) subspace reducing memory overhead (Fig. 1).

The primary contributions of this work are:

1. **Scalable Training-free NAS Framework:** We introduce TraceNAS, a unified pruning framework (Fig. 2) for joint search of LLM depth and width. Our approach enables zero-shot discovery of non-uniform pruned models, facilitating efficient compression across model scales without search-time training.
2. **Zero-shot Proxy for Functional Alignment:** We introduce a scale-invariant, gradient-based zero-shot proxy that evaluates gradient alignment between the pretrained and pruned models. Our proxy achieves 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 7.5×, 1.35× and 10× reduction in search-time wall-clock hours, peak memory and total calibration data, respectively, compared to training-based search baselines (Fig. 1). This drastically lowers the overhead for large-scale model compression by ensuring search costs are significantly lower than recovery training.

[1]TraceNAS, DarwinLM, ShearedLlama, EvoPress, Flextron, DISP-LLM and LLMPruner report search time on NVIDIA A100 and we report original E^3 search time in NPU hours. Accuracy values reported are for Llama2-7B.

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 Training-aware vs. Zero-shot LM Pruning

Discovering non-uniform architectures for structured pruning typically yields better performance than uniform sparsity (Xia et al., 2023; Tao et al., 2023). However, exploring this space via training-aware search (DarwinLM (Tang et al., 2025), PUZZLE (Bercovich et al., 2025)) incurs massive computational costs that often rival the post-pruning recovery training itself. As models scale, training-free model discovery via Neural Architecture Search (NAS) becomes necessary. Standard zero-shot proxies (Abdelfattah et al., 2021; Li et al., 2023) fail for pruned models because they evaluate randomly initialized networks and ignore pretrained knowledge inheritance. While recent zero-shot pruning methods like LPZero (Dong et al., 2024a) and Pruner-Zero (Dong et al., 2024b) address this, they rely on an overhead-heavy, dual-stage metric search. Furthermore, EvoPress (Sieberling et al.) employs evolutionary search for heterogeneous discovery of pruned LLMs guided by KL divergence and loss of candidate models. In contrast, TraceNAS introduces a unified, training-free fitness proxy that directly measures a sub-network's functional alignment with its pretrained base, streamlining the discovery of highly performant architectures with fast recovery convergence.

We provide an extended related works section in Appendix A.

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 our training-free proxy $\Phi(\cdot,\cdot)$. Φ measures functional alignment, the ability of $\mathcal{M}_{sub}$ to recover performance by preserving the gradient direction of $\mathcal{M}_{base}$ on its loss landscape:

$$\hat{\mathcal{M}}_{sub} = \arg\max_{\mathcal{M}\in\mathcal{G}} \Phi(\mathcal{M}_{sub}, \mathcal{M}_{base}) \text{ s.t. } \mathcal{P}(\mathcal{M}_{sub}) \leq C, \tag{1}$$

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 $(\mathbf{d}, \boldsymbol{\kappa})$. 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

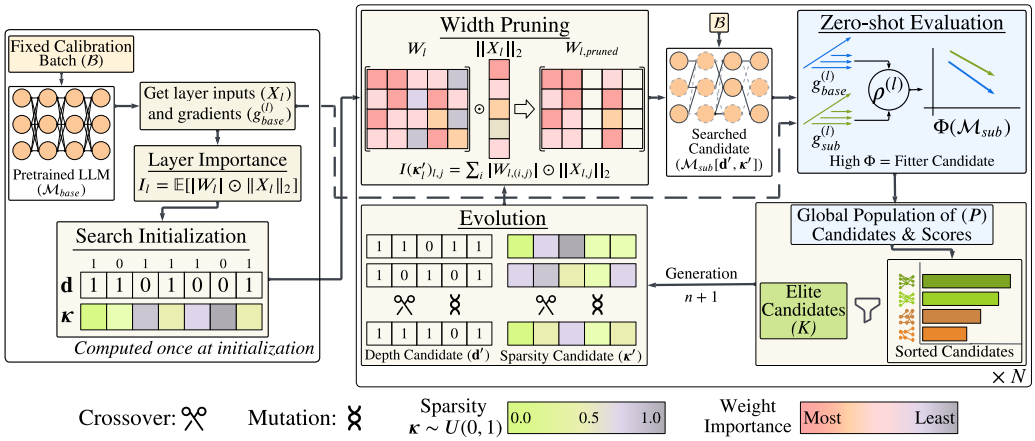


Figure 2: **Visualization of the TraceNAS search framework.** TraceNAS uses a gradient-based, training-free proxy to guide structural pruning. Following a one-time initialization of base gradient traces (g_{base}) and importance scores (I_l), a population of depth ($\mathbf{d}$) and width ($\boldsymbol{\kappa}$)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

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 $\boldsymbol{\kappa} = \{\kappa_1, \ldots, \kappa_L\}$ where $\kappa_l = (r^{(l)}_{attn}, r^{(l)}_{mlp}) \in (0, 1]^2$ 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 $\boldsymbol{\kappa}$ to binary masks and implementing them via in-place masking.

Width Mask Generation To realize a layer's width configuration κ_l into a weight mask, we use the activation-weighted product (Sun et al., 2023):

$$I(\kappa_l)_{l,j} = \sum_i |W_{l,(ij)}| \cdot \|X_{l,j}\|_2 \tag{2}$$

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. We evaluate complementary heuristics in Appendix. B.5.

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 alignment and recovery potential of candidate models $\mathcal{M}_{sub}$, this section defines 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).

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 preservation. 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 computational 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 allows us to capture global dependencies while significantly reducing the memory footprint of gradient storage.

Measuring Functional Alignment To quantify how effectively $\mathcal{M}_{sub}$ aligns with 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:

$$\rho^{(l)} = \frac{1}{N_l} \left\langle \left(\frac{g^{(l)}_{sub} - \mu_{g^{(l)}_{sub}}}{\sigma_{g^{(l)}_{sub}}} \right), \left(\frac{g^{(l)}_{base} - \mu_{g^{(l)}_{base}}}{\sigma_{g^{(l)}_{base}}} \right) \right\rangle \tag{3}$$

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 $\bar{r}^{(l)} = 1 - r^{(l)} \in \kappa_l$:

$$\Phi(\mathcal{M}_{sub}, \mathcal{M}_{base}) = \sum_{l\in\text{Attn}} \bar{r}^{(l)}_{attn} \cdot \rho^{(l)} + \sum_{l\in\text{MLP}} \bar{r}^{(l)}_{mlp} \cdot \rho^{(l)} \tag{4}$$

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 alignment. By weighting $\rho^{(l)}$ by parameter density, we prevent Φ from being skewed by the high-variance noise typical of aggressively pruned sub-blocks. We provide a detailed overview of our selection and evolutionary mechanics in Appendix B.1.

3.5 INTERPRETING GRADIENT TRACE ALIGNMENT

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

- **Gradient Anchoring:** High-performing pretrained models reside within flat, stable convergence basins (Frankle et al., 2020; Li et al., 2018). Our design is motivated by the intuition

that a high alignment score Φ reflects the extent to which the per-layer gradient of a candidate's remains anchored with that of the base model. By preserving the gradient's direction, $\mathcal{M}_{sub}$ is intended to stay aligned with the original model. This framing is consistent with the sustained rank correlation we observe between Φ and downstream accuracy throughout CPT (Appendix D.4).

- **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's optimization. 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. We verify this empirically in Appendix D.5.
- **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 also stabilize activation scales across the depth of the model, ensuring that Φ captures the underlying optimization trajectory rather than numerical 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-Instruct (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 are provided in Appendix E.3 and E.4, respectively.

Implementation Details The evolutionary search uses a population of 30 candidates over 50 search iterations. On a single NVIDIA H200 GPU, the search concludes in $\sim$2 hours, a significant acceleration over the 8.5-hour A100 baseline reported in Fig. 1. The highest scored sub-networks then undergo CPT on a cluster of six H200 GPUs ($\sim$24 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 4096 for Llama-2 and Qwen-2.5, and 8192 for Llama-3.1. Detailed hyperparameters in Appendix B.6.

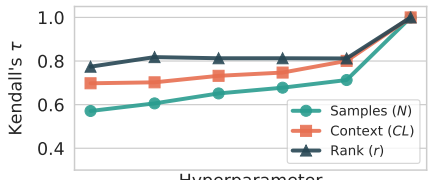
Baselines We benchmark TraceNAS against leading structured and layer-pruning methods. This includes, mask-learning methods, ShearedLlama (Xia et al., 2023) and E^3-Pruner (Yuan et al., 2025), which employ Lagrangian optimization. We compare performance with LLMPruner (Ma et al., 2023) which uses Taylor 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. Additionally, to compare TraceNAS against other training-free search methods we evaluate EvoPress (Sieberling et al.). 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) are included in Appendix E.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 Wikitext-2 (Merity et al., 2016) perplexity (PPL), 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).

Method	PPL		MMLU		Avg. Acc.	
	SP	KT	SP	KT	SP	KT
#Params	0.87	0.72	0.11	0.07	0.66	0.50
Loss	0.91	0.77	0.49	0.31	0.85	0.71
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 – Isolated	0.68	0.57	0.43	0.26	0.50	0.40
TraceNAS	**0.93**	**0.79**	0.54	0.39	**0.94**	**0.82**

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



N Samples	4	8	16	64	128	256
CL Tokens	128	256	512	1024	2048	4096
Rank *r*	2	8	64	256	1024	4096

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

4.2 TraceNAS Proxy Validation via Ranking Correlation

To evaluate TraceNAS as a proxy for recovery potential, 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 CPT setup. We compute Spearman Rho (SP) and Kendall Tau (KT) correlations against PPL, MMLU and average downstream accuracy after Once-for-All (OFA) (Cai et al., 2019; Sukthanker et al.) finetuning on Alpaca (Taori et al., 2023).

Correlation with Downstream Performance NASWOT (Mellor et al., 2021) and ZiCo (Li et al., 2023) rank models based on their deature-based expressivity and convergence capability, based on loss-landscape smoothness, from random initializations. Furthermore, #Params and ZiCo achieve identical correlations because they induce the same ranking over the 70 sampled models. ZiCo sums per-parameter mean-to-standard-deviation gradient ratios across the network; for a converged pretrained model these ratios are small and nearly uniform, so the score is dominated by the number of parameters it aggregates over. This effectively leads to a random search across parameter scales, since they upweight larger models. 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. Loss and MeCo (Jiang et al., 2023) are both forward-only metrics and perform well on PPL, however, they fail to predict 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.

In contrast TraceNAS explicitly accounts for alignment between the pretrained base and pruned models. We evaluate the Pearson correlation of gradients between the base model and each sampled subnetwork, weighted by layer-wise retention (Φ). By measuring this alignment, Φ captures how structural changes impact the calibration gradients of the sampled model. This allows the proxy to account for dependencies that feature-based or loss-landscape smoothness-based metrics fail to capture. TraceNAS maintains the highest combined ranking across all three evaluation metrics because its gradient centering specifically isolates functional signals from directional noise. Furthermore, 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, unweighted aggregation of $\rho^{(l)}$ coefficients and a layer-isolated variant.

A detailed analysis of ranking correlation and hyperparameter stability is provided in Appendix D.1 and D.3. We provide isolated zero-shot proxy evaluations on Llama-2 7B in Appendix D.5 and CPT ranking correlations across 30 randomly sampled architectures in Appendix D.4.

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 aligns 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.

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.

Table 2: Pruning results for Llama-2 7B. The top section reports scores from the original papers or from re-evaluated open-source checkpoints. The bottom section reports reproduced models (†), each recovered with 20B tokens of CPT using a WSD schedule. Srch and Recv denote tokens used during search and recovery and averages include MMLU where available.

Method	#Params	Srch	Recv	PPL (↓)	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg. (↑)
Llama-2 7B	6.7B	–	–	5.47	46.53	77.69	74.11	76.34	52.81	78.96	93.80	29.80	77.73	67.53
Flextron[3]	3.4B	1.0K	90B	–	36.80	74.10	62.00	66.50	–	68.50	–	–	–	–
LoRAP	2.7B	16.0K	50K steps	32.16	24.69	57.20	47.90	31.30	26.30	30.00	51.20	27.50	61.90	39.78
E^3	2.7B	0.5B	–	–	–	71.20	57.30	65.40	38.10	54.60	88.40	28.10	63.60	58.34
Uniform	3.4B	98.0M	20B†	17.97	24.01	65.10	58.30	69.50	38.20	54.30	87.10	26.03	59.90	53.60
LLMPruner	2.9B	1.3K	20B†	13.57	24.54	68.43	59.52	65.52	36.84	59.73	88.71	26.76	59.21	54.36
EvoPress	2.7B	26.5M	20B†	11.18	24.39	70.83	58.16	65.61	37.96	60.73	89.00	29.33	63.73	55.52
DISP-LLM	2.7B	–	20B†	13.32	24.86	70.83	62.42	68.52	39.75	65.87	89.21	27.98	60.52	56.66
ShearedLlama	2.7B	0.4B	20B†	12.24	24.76	72.92	60.80	67.46	38.06	64.73	90.30	28.72	63.91	56.85
DarwinLM	2.7B	1.0B	20B†	11.22	25.28	73.17	63.59	69.90	42.91	66.57	89.00	29.18	**63.63**	58.14
TraceNAS	2.7B	98.0M	20B†	**10.02**	**25.37**	**73.40**	**63.97**	**70.20**	**43.80**	**68.54**	**91.10**	**29.71**	61.76	**58.65**

Performance on Llama-2 7B As shown in Table 2, TraceNAS matches or exceeds all baselines in average accuracy at a fraction of their search cost. For a fair comparison, we reimplement and retrain the pipelines and models reported by DarwinLM, ShearedLlama, DISP-LLM, EvoPress and LLMPruner on our 20B-token CPT setup (†). Among training-free methods, LLM-Pruner trails substantially, and EvoPress, which also uses evolutionary search but scores candidates by calibration loss and KL divergence, trails TraceNAS by 3.13% in average accuracy. This suggests loss-based fitness preserves language modeling but not downstream capability, consistent with our isolated proxy

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

[3] Flextron (Cai et al., 2024) and Minitron (Sreenivas et al., 2024) serve as accuracy upper bounds rather than direct efficiency competitors. They utilize ∼90B and ∼94B recovery tokens (∼4.5× and ∼4.7× TraceNAS's budget), respectively.

Table 3: Pruning results for Llama-3.1 8B and Qwen-2.5 14B-Instruct. E^3 and Minitron do not have public code, so we do not run them for Llama-3.1 and Qwen-2.5, respectively. Average includes MMLU.

Method	#Params	Srch	Recv	PPL (↓)	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg. (↑)
Llama-3.1 8B	8B	–	–	6.23	64.78	81.22	77.42	81.64	57.59	81.81	96.20	31.02	82.14	72.65
Minitron[3]	4.4B	–	94B	7.29	60.50	76.82	73.50	76.89	56.14	76.03	95.23	30.02	83.21	69.81
Uniform (Ours)	4.6B	98.0M	20B†	29.35	28.76	68.32	58.54	68.23	47.89	66.42	88.48	27.54	66.59	57.86
LLMPruner	4.6B	1.3K	20B†	32.53	29.45	54.76	48.93	45.16	32.04	33.81	76.8	24.42	59.63	45.00
DarwinLM	4.6B	1.0B	20B†	30.19	**43.32**	**74.53**	65.26	73.13	51.62	**71.17**	93.24	30.76	70.08	**63.68**
TraceNAS (Ours)	4.6B	98.0M	20B†	**14.32**	30.12	74.23	**66.08**	**73.50**	**52.26**	69.01	**93.90**	**30.81**	**73.00**	62.55
Qwen-2.5 14B	14B	–	–	5.29	82.53	82.10	79.71	85.81	72.44	85.10	96.70	41.16	87.95	79.28
E^3	8.2B	0.5B	–	–	36.50	76.90	63.00	76.00	47.90	67.00	93.70	30.00	66.50	61.94
Uniform (Ours)	8.4B	98.0M	20B†	29.89	51.23	71.32	62.64	77.12	50.02	69.89	88.91	28.53	69.45	63.23
LLMPruner	8.4B	1.3K	20B†	22.54	30.72	51.30	50.19	43.84	37.30	45.56	75.6	26.04	52.23	45.86
DarwinLM	8.4B	1.0B	20B†	28.87	56.97	**78.17**	72.09	79.86	57.03	**74.96**	90.32	**32.80**	73.17	68.37
TraceNAS (Ours)	8.4B	98.0M	20B†	**13.72**	**58.79**	76.63	**73.24**	**81.07**	56.05	73.77	**95.70**	32.56	**73.70**	**69.06**

comparison (Appendix D.5). Training-based methods (DISP-LLM, E^3, LoRAP) also underperform despite costlier pipelines, and ShearedLlama degrades when restricted to 20B tokens, reflecting its reliance on high-token recovery. TraceNAS further outperforms the larger 3.4B uniform baseline and is comparable to Flextron, which uses $4.5\times$ more recovery tokens. DarwinLM performs similarly to TraceNAS but requires multi-stage search, $10.2\times$ more search tokens and is further followed by 20B tokens of CPT.

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, TraceNAS significantly improves on the uniform baseline and outperforms DarwinLM across reasoning tasks while being outperformed on MMLU solely. 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.

On Qwen-2.5, LLM-Pruner (Ma et al., 2023) exhibits performance degradation and TraceNAS surpasses both DarwinLM and E^3-Pruner. Notably, across all architectures, TraceNAS produces sub-networks with high recovery potential and are discovered at a fraction of the computational search cost required by existing training-aware methods. Detailed inference, search, and recovery speedup analysis against all publicly available baselines are provided in Appendix E.1, and E.2.

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 matched performance across architectures at a fraction of the overhead compared to training-aware search 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.

AI use statement

In this paper, we use an LLM to help polish the writing of the paper. All ideas, experiments and references have been carried out by the authors.

Ethics statement

The research presented in this paper advances the accessibility of high-performing LLMs by significantly lowering the computational barriers to large-scale model compression. By providing a

training-free framework for architectural discovery, we enable the development of efficient models even under resource constraints. Furthermore, the substantial reduction in GPU-hours required for model search directly mitigates the carbon footprint and energy consumption associated with large-scale NAS. These advancements enable the deployment of LLMs in edge environments, supporting sustainable AI development.

Reproducibility statement

We provide detailed descriptions of our algorithms, search and CPT hyperparameters and our anonymized code can be found here: https://anonymous.4open.science/r/TraceNAS-anon-CD9A.

References

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021.

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. Slicegpt: 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.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*, 2023.

Peijie Dong, Lujun Li, Xiang Liu, Zhenheng Tang, Xuebo Liu, Qiang Wang, and Xiaowen Chu. Lpzero: Language model zero-cost proxy search from zero. *arXiv preprint arXiv:2410.04808*, 2024a.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. *arXiv preprint arXiv:2406.02924*, 2024b.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Yonggan Fu, Zhongzhi Yu, Junwei Li, Jiayi Qian, Yongan Zhang, Xiangchi Yuan, Dachuan Shi, Roman Yakunin, and Yingyan Celine Lin. Amoeballm: Constructing any-shape large language models for efficient and instant deployment. *Advances in Neural Information Processing Systems*, 37:78299–78319, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024a. URL https://zenodo.org/records/12608602.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244, 2024b.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers, 2024. *URL https://arxiv.org/abs/2403.17887*, 2024.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Thorir Mar Ingolfsson, Mark Vero, Xiaying Wang, Lorenzo Lamberti, Luca Benini, and Matteo Spallanzani. Reducing neural architecture search spaces with training-free statistics and computational graph clustering. In *Proceedings of the 19th ACM International Conference on Computing Frontiers*, pp. 213–214, 2022.

Tangyu Jiang, Haodi Wang, and Rongfang Bie. Meco: zero-shot nas with one data and single forward pass via minimum eigenvalue of correlation. *Advances in Neural Information Processing Systems*, 36:61020–61047, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11:1, 2024.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier dimensions that disrupt transformers. *arXiv preprint arXiv:2105.06990*, 2021.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*, 2023.

Guangyan Li, Yongqiang Tang, and Wensheng Zhang. Lorap: Transformer sub-layers deserve differentiated structured compression for large language models. *arXiv preprint arXiv:2404.09695*, 2024.

Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. *arXiv preprint arXiv:2301.11300*, 2023.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Lujun Li and Zhe Jin. Shadow knowledge distillation: Bridging offline and online knowledge transfer. *Advances in Neural Information Processing Systems*, 35:635–649, 2022.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Modegpt: Modular decomposition for large language model compression. *arXiv preprint arXiv:2408.09632*, 2024.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 467–484, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ziyang Luo, Artur Kulmizev, and Xiaoxi Mao. Positional artefacts propagate through masked language model embeddings. In *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. 5312–5327, 2021.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International conference on machine learning*, pp. 7588–7598. PMLR, 2021.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5):1, 2023.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.

Anthony Sarah, Sharath Nittur Sridhar, Maciej Szankin, and Sairam Sundaresan. Llama-nas: Efficient neural architecture search for large language models. In *European Conference on Computer Vision*, pp. 67–74. Springer, 2024.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. Evopress: Accurate dynamic model compression via evolutionary search. In *Forty-second International Conference on Machine Learning*.

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. ISSN 00029556. URL http://www.jstor.org/stable/1412159.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, et al. Llm pruning and distillation in practice: The minitron approach. *arXiv preprint arXiv:2408.11796*, 2024.

Rhea Sanjay Sukthanker, Benedikt Staffler, Frank Hutter, and Aaron Klein. Large language model compression with neural architecture search. In *Workshop on Machine Learning and 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.

Hidenori 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.

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.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai

Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

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

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

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

Tao Yuan, Haoli Bai, Yinfei Pan, Xuyang Cao, Tianyu Zhang, Lu Hou, Ting Hu, and Xianzhi Yu. E ^{3}-pruner: Towards efficient, economical, and effective layer pruning for large language models. *arXiv preprint arXiv:2511.17205*, 2025.

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

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

APPENDIX

TABLE OF CONTENTS

A Extended Related Work **16**
A.1 Influence-based Importance . . . 16
A.2 Training-Aware LM Pruning . . . 17
A.3 Zero-shot NAS for LM Pruning . . . 17

B Implementation & Search Analysis **17**
B.1 Selection & Evolutionary Mechanics . . . 18
B.2 Evolutionary Search Initialization . . . 18
B.3 Search Space Analysis . . . 19
B.4 Performance Before Post-pruning CPT . . . 19
B.5 Performance Across Varied Search Constraints . . . 20
B.6 Evolution Search and CPT Hyperparameters . . . 20
B.7 Pseudocode . . . 21

C Motivation & Conceptual Framework **21**
C.1 Influence Functions . . . 21
C.2 Structural Sensitivity via Data-Centric Influence . . . 22
C.3 TracIn & the Link to TraceNAS . . . 22
C.4 Why this works . . . 23

D Validating the TraceNAS Proxy (Φ) **23**
D.1 Correlation with Downstream Performance . . . 23
D.2 Validating the Proxy Formulation . . . 24
D.3 Robustness of Proxy Stability: Correlation with Downstream Accuracy . . . 25
D.4 Correlation with Post-pruning CPT . . . 26
D.5 End-to-end Isolated Proxy Comparison . . . 26
D.6 Performance Across Different Sparsities . . . 27

E Scalability & Speedup Analysis **27**
E.1 Inference Speedup Analysis . . . 28
E.2 Search and Recovery Speedup Analysis . . . 29
E.3 Extending TraceNAS to Different Model Scales . . . 29
E.4 Training with more tokens . . . 30
E.5 Evaluating TraceNAS for Unstructured Pruning . . . 30

F Generative Capabilities **31**
F.1 Evaluation on GSM8K . . . 31
F.2 Qualitative Example . . . 31

A EXTENDED RELATED WORK

A.1 INFLUENCE-BASED IMPORTANCE

Beyond channel-wise heuristics (Sun et al., 2023; An et al., 2024), 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 TracIn (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 C.

A.2 Training-Aware LM Pruning

Recent structured pruning methods, such as ShearedLlama (Xia et al., 2023) and E^3 (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. SIMPLE (Tao et al., 2023) employs 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.

A.3 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. Furthermore, EvoPress (Sieberling et al.) employs evolutionary search for heterogeneous discovery of pruned LLMs guided by KL divergence and loss of candidate models. 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.

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 influence long-term convergence. Its performance is not uniformly superior to training-aware search: on Llama-3.1-8B, TraceNAS trails DarwinLM on MMLU, consistent with Φ being least predictive of MMLU in Table 1. We hypothesize that factual knowledge retention is particularly sensitive to width pruning in GQA models. 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 Implementation & Search Analysis

In this section we provide implementation details; the evolutionary search mechanics, search initialization and analysis of search dynamics. We also provide zero-shot pruned model performance, varied search constraints and width pruning heuristics, search space and CPT hyperparameters and the TraceNAS pseudocode.

B.1 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 B.2). A hybrid strategy optimizes the joint $(\mathbf{d}, \boldsymbol{\kappa})$ 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.

B.2 Evolutionary Search Initialization

Table 4: Comparison of Llama-2 7B pruning using evolutionary search and importance based initialization, highlighted as TraceNAS (Ours). MMLU is not reported as all models achieve 25% accuracy (which is essentially random).

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	58.42	51.86	46.72	37.13	48.45	78.79	23.85	56.98	50.28
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	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 $I_l = \mathbb{E}\big[|W_l| \odot \|X_l\|_2\big]$, which characterizes the sensitivity of each layer.

These importance scores define a non-uniform sampling prior over the sub-module width sparsity configurations $\boldsymbol{\kappa} = \{\kappa_1, \ldots, \kappa_L\}$, where $\kappa_l = (r^{(l)}_{attn}, r^{(l)}_{mlp})$. 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 12.53% 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 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.

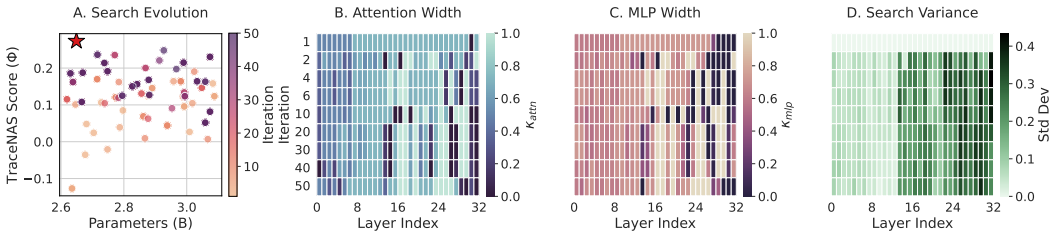


Figure 4: **TraceNAS Evolutionary Dynamics and Search Convergence.** Search trajectory for pruning Llama-2-7B to 2.7B across 50 iterations. (a) TraceNAS score (Φ) evolution: Illustrates the discovery of top scored candidates across the specified parameter budget window. The red star indicates the model with maximal alignment for the given constraint. (b) Attention Width Evolution: Tracks the sparsity ratios for the attention sub-block $(r_{attn}^{(l)})$; the search identifies specific layers where attention heads are critical for maintaining representational flow. (c) MLP Width Evolution: Sparsity ratios for MLP sub-blocks $(r_{mlp}^{(l)})$, 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), $\kappa = 1$ indicates the layer is fully pruned.

B.3 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 alignment, 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 distinct sparsity profiles for attention $(r_{\text{attn}}^{(l)})$ and MLP $(r_{\text{mlp}}^{(l)})$ widths. The search consistently preserves the final MLP layers, indicating that they are critical to representational flow, attention modules in earlier layers are pruned more aggressively, suggesting lower sensitivity to attention capacity at depth. 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. Variance is low in the first iteration because the initial population is sampled from a concentrated importance prior, and it expands as crossover and mutation introduce diversity. The subsequent plateau indicates that the search converges to a stable set of high-scoring architectures within the iteration budget.

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	2.7B	98M	–	61.12	51.40	43.33	31.03	42.35	47.5	25.96	50.55	44.16

B.4 Performance Before Post-pruning CPT

We provide zero-shot results after pruning with all previously reported baselines and additional methods in Table 5. 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 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 Sec. E.2 and Table 12.

B.5 Performance Across Varied Search Constraints

To showcase the modularity of TraceNAS we compare layer-only pruning to full layer and width pruning using TraceNAS in Table 6. 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.
Llama-2 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	Wanda	2.7B	**73.40**	**63.97**	70.20	43.80	**68.54**	**91.10**	**29.71**	61.76	**62.81**

B.6 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 each LoRA adapter $\Delta W = BA$, with $B \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A \in \mathbb{R}^{r \times d_{\text{in}}}$, using the standard scheme: A is drawn from a Kaiming Normal distribution (He et al., 2015) and B is initialized to zero. Since $BA = 0$, the adapters leave the forward pass of $\mathcal{M}_{base}$ unchanged. Moreover, because $\nabla_A \mathcal{L} = B^\top G = 0$, only the gradients with respect to B are non-zero, given by $\nabla_B \mathcal{L} = GA^\top$, where $G = \nabla_W \mathcal{L}$ is the full-rank weight gradient. The low-rank gradient trace is therefore a random projection of the full weight gradient onto an r-dimensional subspace. We use the same fixed A matrices (a shared random seed) for $\mathcal{M}_{base}$ and every candidate $\mathcal{M}_{sub}$, so all gradient traces are projected onto a common subspace and remain directly comparable.

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 (T)	50			Context Length	4,096	8,192	4,096
–	–			Overall Tokens	20B		

B.7 PSEUDOCODE

Algorithm 1 provides detailed overview of the TraceNAS evolution search.

Algorithm 1: TraceNAS

Input : Pretrained $\mathcal{M}_{base}$; Budget C; Calibration set $\mathcal{B}$; Sparsity $\boldsymbol{\kappa}$; 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`
$I_l \leftarrow \mathbb{E}\big[|W_l| \odot \|X_l\|_2\big]$; `// Compute block importance for all` L

Initialize P_0 with importance-score based initialization $\mathbf{d} \in \{0,1\}^L$ and width $\boldsymbol{\kappa}$ weighted by I_l;
for $t = 1$ **to** T **do**
  for *each candidate* $(\mathcal{M}_{sub}(\mathbf{d}, \boldsymbol{\kappa})) \in P_t$ **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**
      $\rho^{(l)} \leftarrow$ Pearson Correlation Coefficient$(g_{sub}^{(l)}, g_{base}^{(l)})$; `// Gradient Trace Correlation`
    `// Fitness Aggregation`
    $\Phi(\mathcal{M}_{sub}) \leftarrow \sum_{l\in\text{Attn}} \bar{r}_{attn}^{(l)} \rho^{(l)} + \sum_{l\in\text{MLP}} \bar{r}_{mlp}^{(l)} \rho^{(l)}$;
  `// Elite Selection and Reproduction`
  $E_t \leftarrow$ Select top k candidates from P_t based on Φ;
  $P_{t+1} \leftarrow E_t$; `// Carry over elite set directly`
  while $|P_{t+1}| < |P_t|$ **do**
    Parent $A, B \leftarrow$ Sample from E_t;
    Child $C \leftarrow$ Crossover(A, B) + Mutation(C);
    Add Child C to P_{t+1};
return $\hat{\mathcal{M}}_{sub} \leftarrow \arg\max_{\Phi}\{P_T\}$

C MOTIVATION & CONCEPTUAL FRAMEWORK

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

C.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:

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

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

C.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:

$$\begin{aligned} I^{(l)}(z_i) = -\sum_{j=1}^{m} \nabla_\theta^{(l)} \mathcal{L}(z_j^V, \theta)^\top \\ \cdot \left[H^{(l)}(\theta) \right]^{-1} \nabla_\theta^{(l)} \mathcal{L}(z_i, \theta) \end{aligned} \tag{5}$$

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

C.3 TracIn & the Link to TraceNAS

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

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

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

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

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

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

C.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 GradNorm 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 Alignment, whether the pruned model aligns with 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.

D VALIDATING THE TRACENAS PROXY (Φ)

This section provides a detailed analysis of the TraceNAS proxy Φ and its ranking correlation with PPL, MMLU, and average downstream accuracies compared against the established baselines in Table 1. We further validate the formulation of our proxy and analyze its robustness to the search-space hyperparameters. Finally, we validate that TraceNAS can model post-pruning CPT performance throughout the training cycle, perform an end-to-end proxy comparison on a unified search and CPT setup and validate performance across different target sparsity.

D.1 CORRELATION WITH DOWNSTREAM PERFORMANCE

Expressivity & Convergence Proxies: As shown in Table 1, TraceNAS achieves superior ranking correlation with downstream performance by effectively aligning with the pretrained base. 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 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 alignment that simpler magnitude-based metrics miss and functions as a reliable proxy for model performance.

D.2 Validating the Proxy Formulation

To justify the formulation of Φ, we evaluate several design choices within our low-rank gradient setup: the similarity metric, the aggregation strategy, and the gradient source.

First, we compare similarity metrics. We observe that cosine similarity (TraceNAS – Cosine) is more robust than the dot product (TraceNAS – Dot), which fails because it lacks the scale-invariance necessary to handle the significant magnitude shifts inherently caused by structural pruning. However, cosine similarity remains sensitive to mean-gradient bias. The Pearson correlation coefficient addresses this by centering the gradient traces.

Second, we evaluate the aggregation strategy. Uniformly aggregating these coefficients across layers (TraceNAS – Unweighted) introduces instability in highly compressed blocks. By using layer width retention as a weighting factor, Φ acts as a dynamic noise-filtering mechanism, de-emphasizing heavily pruned layers. This weighting anchors the global score in high-capacity regions and prevents the global fitness signal from being skewed by the high variance typical of aggressively pruned, low-capacity blocks.

Finally, to demonstrate the necessity of modeling cross-layer error propagation, we evaluate a strictly layer-local ablation (TraceNAS – Isolated). In this variant, we replace the global loss gradient with a per-layer reconstruction loss, computing gradients for each block in isolation. As shown in Table 1, removing the global signal causes a severe drop in rank correlation (Kendall τ for average accuracy drops from 0.82 to 0.40). This sharp degradation confirms that layer-local heuristics are insufficient for evaluating how pruning errors compound throughout the network, underscoring the necessity of TraceNAS's global end-to-end gradient trace.

Table 8: Spearman ρ correlation between various search hyperparameters and model PPL and average downstream accuracy. All correlation values are averaged over 3 random seeds. Best value per column in **bold**; ‡ denotes the setting used in our search.

(a) N			(b) CL			(c) r		
N	PPL	Avg Acc	CL	PPL	Avg Acc	r	PPL	Avg Acc
4	0.893	0.912	128	**0.932**	0.910	2	0.923	0.905
8	0.925	0.921	256	0.915	0.928	8	**0.933**	0.918
16‡	**0.930**	**0.941**	512	0.916	0.921	64‡	0.930	**0.941**
64	0.923	0.938	1024	0.930	**0.941**	256	0.928	**0.941**
128	0.928	0.924	2048	0.915	0.936	1024	0.928	0.929
256	0.901	0.897	4096‡	0.930	0.939	4096	0.929	0.931

Table 9: Kendall τ correlation between various search hyperparameters and model PPL and average downstream accuracy. All correlation values are averaged over 3 random seeds. Best value per column in **bold**; ‡ denotes the setting used in our search.

(a) N			(b) CL			(c) r		
N	PPL	Avg Acc	CL	PPL	Avg Acc	r	PPL	Avg Acc
4	0.742	0.780	128	0.783	0.738	2	0.789	0.785
8	0.790	0.789	256	0.779	0.779	8	0.787	0.801
16‡	**0.791**	**0.821**	512	0.750	0.781	64‡	0.791	**0.821**
64	0.778	0.808	1024	0.791	**0.821**	256	0.790	0.819
128	0.790	0.818	2048	0.767	0.806	1024	**0.792**	0.812
256	0.787	0.819	4096‡	**0.797**	0.818	4096	0.789	0.818

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

As shown in Sec. 4.2, the TraceNAS proxy (Φ) produces rankings that remain stable across search hyperparameters. To verify that this stability also holds with respect to ground-truth performance, we measure Spearman ρ and Kendall τ correlations between proxy scores computed under each hyperparameter setting and WikiText-2 perplexity (PPL) and average downstream accuracy (Tables 8 and 9).

Across all configurations, Φ maintains a Spearman correlation of at least 0.89 with average accuracy, and our search settings achieve the highest or near-highest correlation in every column. We analyze each hyperparameter below, reporting values Spearman ρ values:

- **Calibration Sample Density (N):** With as few as 4 calibration samples, the proxy achieves a correlation of 0.912 with average accuracy. Correlation peaks at $N = 16$ (0.941), the setting used in our search, and remains stable through $N = 128$.
- **Context Length (CL):** Correlations remain high even at short context windows; at $CL = 128$, the proxy achieves 0.932 with PPL and 0.910 with average accuracy. Accuracy correlation improves with longer contexts, peaking at $CL = 1024$ (0.941) and remaining stable at $CL = 4096$ (0.939). This indicates that while local dependencies are captured early, longer contexts improve the proxy's ability to predict downstream reasoning performance.
- **Low-Rank Gradient Subspace (r):** The proxy is highly stable across gradient subspace ranks. Even at $r = 2$, correlation with average accuracy remains at 0.905. Correlation peaks at $r = 64$ (0.941), the rank used during search, and does not improve at full rank ($r = 4096$: 0.931), confirming that low-rank gradient traces are sufficient for performance ranking while minimizing memory overhead.

These results show that Φ provides a reliable ranking signal under constrained search budgets, supporting the use of small calibration sets, moderate context lengths, and low-rank gradient subspaces during search.

D.4 Correlation with Post-pruning CPT

To validate the reliability of our zero-shot proxy during the post-pruning recovery phase, we evaluate its rank correlation with downstream model performance across successive checkpoints of CPT (Fig. 5). We randomly sampled 30 architectures from the OFA search space, each starting from pretrained initialization. These models were trained on the FineWeb-Edu dataset for 2.5 billion tokens. We then computed the Kendall τ rank correlation between each zero-shot proxy score evaluated at initialization and the actual model perplexity (PPL) measured at regular CPT step intervals, averaging the results across three independent seeds.

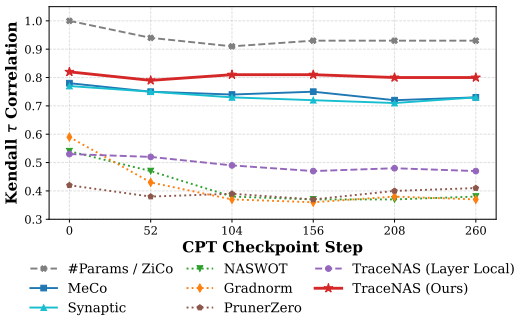


Figure 5: **CPT checkpoint correlations.**

As shown in Fig. 5, while baselines such as ZiCo and raw parameter count (#Params) exhibit high ranking correlation, this is a misleading artifact of size bias. These metrics simply scale with the parameter count and fail to provide meaningful structural guidance during an evolutionary search, as they inherently default to favoring the largest possible model. Specifically, ZiCo is invariant to model width and channel selection, causing its ranking capability to collapse entirely into a proxy for #Params. We further elaborate on these results in Appendix D.5.

In contrast, our proposed TraceNAS proxy demonstrates robust, high correlation ($\tau = 0.82$) directly at pretrained initialization (checkpoint 0) and maintains stability consistently across all recovery checkpoints. Unlike other proxies whose correlations either start weak or degrade as recovery training progresses, TraceNAS reliably isolates architectures with optimal functional alignment and genuine post-recovery performance potential.

Table 10: End-to-end performance of Llama-2 2.7B models discovered by competing zero-shot proxies (trained on 10B CPT tokens). Average includes MMLU.

Method	PPL ↓	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg. ↑
#Params	43.2	23.96	56.25	50.82	47.66	33.45	48.94	53.6	24.04	61.49	44.47
Loss	12.89	24.12	69.65	56.79	64.23	33.89	56.24	84.78	27.74	60.54	53.11
ZiCo	45.17	25.18	57.18	51.22	49.26	31.41	46.14	51.6	23.04	60.97	44.0
Gradnorm	24.06	23.89	58.26	55.90	52.03	34.26	49.75	83.20	24.88	61.98	49.35
Meco	12.05	23.58	69.34	52.64	64.81	34.55	50.21	84.30	24.11	**62.14**	51.74
PrunerZero	13.33	26.11	68.87	53.74	62.58	34.21	47.87	82.90	25.80	62.04	51.57
TraceNAS - Isolated	12.79	**26.30**	70.89	55.48	65.90	36.60	53.68	85.40	26.57	61.37	53.57
TraceNAS	**11.03**	24.35	**71.54**	**59.43**	**68.25**	**40.24**	**66.40**	**89.40**	**28.57**	62.01	**56.69**

D.5 End-to-end Isolated Proxy Comparison

To demonstrate that the TraceNAS proxy directly drives final performance gains, we conduct an isolated end-to-end evaluation comparing sub-networks selected by competing zero-shot proxies. Using identical evolutionary search settings, we prune Llama-2 7B down to a 2.7B target parameter count under each proxy and apply 10B tokens of Continued Pre-Training (CPT) using the exact same recovery setup. As presented in Table 10, the choice of search proxy exerts a profound impact on final model quality post-recovery:

- **Failure of Capacity and Width-Insensitive Metrics:** Relying solely on parameter count (#Params) or proxies that effectively mirror it (ZiCo, which aggregates inverse coefficients of gradients without sensitivity to layer width) results in catastrophic capability collapse. Both metrics yield models with perplexities exceeding 40 and the lowest average downstream accuracies ($\sim$44%), showing that simple structural capacity metrics cannot identify functionally sound configurations.

- **Degradation in Heuristic Proxies:** While gradient-magnitude heuristics like GradNorm show high correlation, the model it identifies exhibits performance drop after CPT, highlighting that gradient norm alone fails to preserve functional alignment.
- **Limitations of Existing Zero-Shot Baselines:** Task-agnostic proxies such as MeCo and PrunerZero yield functional sub-networks, but their lack of direct alignment with pretrained features bounds their post-recovery ceiling.
- **Impact of Full Gradient Alignment:** Comparing TraceNAS against its layer-local variant (TraceNAS – Isolated) underscores the necessity of modeling cross-layer error propagation. Global trace correlation yields a substantial gain of $+3.12\%$ in average accuracy and lowers perplexity. Notable improvements are observed on reasoning-intensive benchmarks such as HellaSwag, Arc-Challenge, and SciQ.

D.6 Performance Across Different Sparsities

We evaluate TraceNAS across a range of sparsity levels and report WikiText-2 perplexity for Llama-2 7B in Fig. 6 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 achieves lower perplexity than competing width and layer pruning baselines.

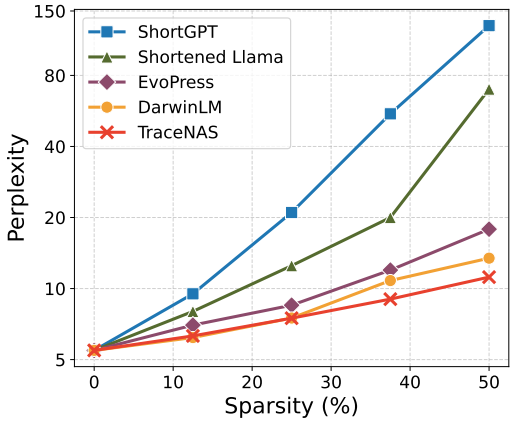


Figure 6: **TraceNAS PPL across sparsity levels.** WikiText-2 perplexity reported for pruned models identified via TraceNAS evolutionary search and trained using 2.5B tokens of CPT.

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 13 for DarwinLM and over 100 for ShortGPT (Men et al., 2025). This result under aggressive compression suggests that the gradient trace proxy effectively identifies architectures that remain aligned with the pretrained model, mitigating representational collapse and preserving language modeling even 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.

E Scalability & Speedup Analysis

In this section we provide inference speedup, end-to-end search and recovery costs and extending the validation of TraceNAS across model scales and variants, training with more CPT tokens and validation on unstructured pruning.

Table 11: Inference 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	671.05	33.24	30.08	7.88	2.18×	1.08×
DarwinLM 2.7B	649.83	**12.96**	**77.16**	**6.07**	2.25×	**2.76**×
TraceNAS 2.7B	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	940.86	34.61	28.89	10.49	1.02×	1.23×
DarwinLM 4.6B	918.68	**18.95**	**52.77**	11.53	1.05×	**2.25**×
TraceNAS 4.6B	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×
DarwinLM 8.4B	1367.75	**46.18**	**21.65**	19.88	1.41×	**1.72**×
TraceNAS 8.4B	1322.30	54.52	18.34	**17.71**	**1.46**×	1.46×

E.1 Inference Speedup Analysis

The practical advantages of the TraceNAS architectures are detailed in Table 11. 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 eager attention and 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 important for interactive responsiveness. Specifically, at the 2.7B scale, TraceNAS reaches a 4.22× prefill speedup (S_p), representing a 1.88× improvement over DarwinLM's prefill speedup. Furthermore, TraceNAS architectures exhibit superior memory efficiency, maintaining the lowest peak memory utilization at the 4.6B and 8.4B parameter scales. For instance, our 8.4B configuration requires only 17.71 GB of peak memory, outperforming 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.

Table 12: Computational Cost Comparison for Llama-2 7B pruning. Search costs are measured by reproducing each baseline's publicly released implementation on our own hardware and search-time peak memory per GPU; recovery costs are reported as a unified training budget (20B tokens) on identical hardware (6× H200 cluster) per model. Both search and recovery time report wall-clock hours. Our reported DarwinLM search time covers Stage 1 (200 generations) only; the full two-stage pipeline is larger for Llama-2 7B.

Method	Search Costs			Recovery Costs		
	Search Hardware	Search Time	Search Peak Memory	Hardware	Tokens	Recovery Time
ShearedLlama	4× A100	63.75 h	18.43 GB	6× H200	20B	24 h
DarwinLM	4× A100	28 h	21.18 GB	6× H200	20B	24 h
EvoPress	1× A100	6 h	14.72 GB	6× H200	20B	24 h
DISP-LLM	4× A100	4 h	19.45 GB	6× H200	20B	24 h
LLM-Pruner	1× A100	10 min	13.62 GB	6× H200	20B	24 h
TraceNAS	1× A100	8.5 h[4]	15.71 GB	6× H200	20B	24 h

E.2 Search and Recovery Speedup Analysis

To ensure a rigorous and transparent comparison of computational costs, we standardize hardware disparities by providing GPU configurations alongside wall-clock time. Quantitative values are depicted in Fig. 1, detailing the search and recovery time, number of GPUs utilized, and the total recovery tokens. To report search overhead we explicitly benchmarked TraceNAS on NVIDIA A100 GPU (yielding an 8.5-hour search time on one GPU) to provide a direct comparison in Table 12.

However, for our main experimental pipeline and the controlled unified recovery setting, all evaluations (including the baseline models we reproduced) were executed on an identical 6× H200 cluster. On this upgraded hardware, the TraceNAS zero-shot search requires only ∼2 hours on a single H200 GPU.

Table 13: Pruning results for Llama-3.1 70B Instruct. Average does not include MMLU.

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	40B	77.21	**82.1**	**85.21**	**83.19**	60.32	78.21	**97.32**	**34.12**	**85.12**	**75.70**

Table 14: Pruning results for Qwen3-8B. Average does not include MMLU.

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	4.1B	65.48	71.78	60.51	70.21	53.28	62.54	93.42	28.35	72.44	64.07

Table 15: Pruning results for Pythia-2.8B and Gemma-2-2B. Average does not include MMLU.

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	59.44
DarwinLM	1.4B	25.1	71.3	57.3	61.2	34.7	54.5	82.9	27.9	65.0	56.85
TraceNAS	1.4B	**25.5**	**71.8**	**58.69**	**62.83**	34.4	54.11	**84.9**	**28.18**	**64.92**	**57.47**
Gemma-2 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	1.2B	**26.44**	**70.34**	**57.93**	**66.70**	**36.77**	**51.84**	**91.2**	**27.49**	**65.62**	**58.49**

E.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 13, 14 and 15. 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) finetuning dataset, further showcasing that TraceNAS can be used as a proxy for finetuning performance.

The benefits of TraceNAS are equally evident in smaller models. For Pythia-2.8B pruned to 1.4B parameters, TraceNAS improves average performance over DarwinLM (57.47 vs. 56.85) and achieves the highest accuracy on most reasoning tasks. For Gemma2-2B pruned to 1.2B parameters, TraceNAS

[4]TraceNAS search time is reported for a single NVIDIA A100 GPU in Table 12 and Fig. 1 to ensure a standardized hardware comparison. For our main experimental pipeline, the search was executed on a single H200 GPU to maximize efficiency.

reduces the performance drop seen in DarwinLM, achieving an average of 58.49 compared to 47.18 while leading across nearly all individual tasks. The results reported for the 2B models are after CPT on FineWeb-Edu 100BT for 10B tokens.

Overall, these results highlight that TraceNAS is robust across scales. By leveraging gradient alignment to guide architectural selection, it consistently produces non-uniform sub-networks that match performance with training-aware search approaches, for both massive 70B models and highly compressed 1.2B models.

Table 16: TraceNAS LLaMA-2-2.7B model trained across 10B, 20B and 50B tokens on FineWeb-Edu 100BT subset. Average includes MMLU.

Method	#Params	Srch	Recv	MMLU	PIQA	WG	ArcE	ArcC	HS	SciQ	LQA	BQ	Avg.
Llama-2 7B	6.7B	–	–	46.53	77.69	74.11	76.34	52.81	78.96	93.80	29.80	77.73	67.53
ShearedLlama	2.7B	0.4B	50B	24.76	**75.80**	64.20	67.00	41.20	**70.80**	90.80	28.20	63.00	58.42
DarwinLM	2.7B	1.0B	10B	25.68	73.12	63.77	69.94	43.94	66.95	90.20	28.57	**64.10**	58.47
TraceNAS	2.7B	98M	10B	24.35	71.54	59.43	68.25	40.24	66.40	89.40	28.57	62.01	56.69
TraceNAS	2.7B	98M	20B	25.37	73.40	63.97	70.20	43.80	68.54	91.10	**29.71**	61.76	58.65
TraceNAS	2.7B	98M	50B	25.87	73.95	**64.35**	**70.93**	**44.67**	69.12	**91.28**	28.18	62.42	58.97

E.4 Training with more tokens

We provide results on training our Llama-2-2.7B pruned model on 10B tokens, 20B tokens and 50B tokens and compare with baselines ShearedLlama and DarwinLM in Table 16.

Table 17: Evaluating Llama-2-7B unstructured pruning using TraceNAS against SOTA unstructured pruning baselines. Average does not include 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
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	50%	38.76	**68.83**	**73.86**	**42.51**	56.42	74.13	**63.15**

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. To apply TraceNAS in a unstructured setting, we employ unstructured weight sparsification within the model and score candidates with Φ by masking out the unimportant neurons. We compare 50% pruned TraceNAS model against magnitude pruning (Han et al., 2015), FLAP (An et al., 2024), Wanda (Sun et al., 2023), AmoebaLLM (Fu et al., 2024) and PrunerZero (Dong et al., 2024b) in Table 17.

We finetune our 50% pruned model on the Orca (Mukherjee et al., 2023) dataset for 10,000 steps with LoRA (Hu et al., 2022), evaluate the public AmoebaLLM SFT checkpoint on Alpaca for 10,000 steps, 50% pruned and finetuned Wanda and FLAP models on Orca. We compare AmoebaLLM in this setting as its use case is a one-for-all trained LLM with multiple small masked LLMs nested within it. 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 17 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.

F GENERATIVE CAPABILITIES

This section validates TraceNAS discovered models by evaluating quantitative and qualitative generative capabilities.

F.1 EVALUATION ON GSM8K

To further highlight the superiority of TraceNAS pruned models in preserving complex reasoning capabilities, we evaluate their generative performance on the GSM8K dataset (Cobbe et al., 2021) under a 5-shot setting.

Fine-Tuning Setup Following the initial post-pruning recovery phase (CPT) on 2.5B tokens, the pruned architectures undergo SFT on the GSM8K training set. We employ LoRA (Hu et al., 2022) with a rank of $r = 64$ and $\alpha = 64$ for efficient parameter updates. SFT is conducted over 3 epochs with a global batch size of 64 using the AdamW optimizer on the base Llama-2 7B model and all pruned models. We apply a peak learning rate of 1×10^{-4} governed by a cosine decay scheduler and a 3% warmup ratio, capping the maximum sequence length at 512 tokens.

Results We present the mathematical reasoning performance for Llama-2 7B models across various pruning ratios in Table 18. For Llama-2 7B, the unpruned base model achieves an accuracy of 26.68%. At a moderate pruning ratio of 15%, TraceNAS recovers an accuracy of 21.75%, outperforming the DarwinLM (19.19%). As the sparsity constraint increases to 35%, downstream reasoning performance naturally degrades across all methods; however, TraceNAS demonstrates notably better knowledge retention, maintaining an accuracy of 12.59% compared to DarwinLM's 8.5%. This performance gap suggests that the sub-networks discovered via our gradient correlation proxy retain superior downstream logical reasoning.

Table 18: GSM8K evaluation on Llama-2 7B across different pruning ratios.

Method	Rank	Pruning Ratio			
		15%	25%	35%	50%
Llama-2 7B	–	26.68			
DarwinLM	64	19.19	17.75	8.5	6.3
TraceNAS		**21.75**	**19.78**	**12.59**	**8.72**

F.2 QUALITATIVE EXAMPLE

To complement our generative benchmark evaluation, we provide a qualitative assessment of the generative capabilities preserved by our pruning framework. Table 19 presents a sample generation from a 60% pruned TraceNAS model (2.7B parameters) given a knowledge-driven prompt. The model adopts the requested persona and produces fluent, multi-paragraph text on the topic. However, it does not fully address the question and introduces fabricated first-person details, reflecting the limits of a heavily compressed model after lightweight recovery without instruction tuning.

Table 19: **Qualitative Analysis:** Sample generation from TraceNAS (2.7B).

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.