Dancing in Fetters: Pareto-Optimal On-Device LLMs via Hardware Co-Design

Luoyang Sun[1,2,3,4*] **Jiwen Jiang**[1,3,4,*] **Yifeng Ding**[4] **Fengfa Li**[4] **Yan Song**[5]
Haifeng Zhang[1,2,‡] **Lei Ren**[4,†] **Kun Zhan**[4] **Chen Wei**[4,†] **Yan Xie**[4] **Jun Wang**[5]
Cheng Deng[6,‡]

[1]Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]AI Lab, The Yangtze River Delta
[4]Li Auto
[5]University College London
[6]The University of Edinburgh

Abstract

Recent embodied AI systems increasingly rely on large language models (LLMs) for high-level planning, semantic reasoning, and long-horizon decision making. However, deploying such capabilities on edge platforms such as autonomous vehicles and mobile robots requires balancing model quality against strict latency and hardware constraints. Existing LLM architectures are primarily designed for cloud-scale accelerators and often become inefficient in on-device settings. We present **PLAS** (**P**areto-optimal **L**LM **A**rchitecture **S**earch), a hardware-aware framework for identifying Pareto-optimal on-device LLM architectures under deployment constraints. Our approach jointly models training loss and inference latency as functions of architectural design choices. We fit an empirical architecture-to-loss model from 170 LLMs trained on 10B tokens each, and estimate latency using roofline-based hardware modeling, enabling efficient exploration of the accuracy–latency Pareto frontier without exhaustively training or profiling every candidate architecture. We profile the inference latency of 1,942 candidate architectures on NVIDIA Jetson Orin. At the same measured latency as Qwen2.5-0.5B, our co-designed architecture achieves 19.42% lower WikiText-2 perplexity. Our results suggest that effective edge deployment requires explicit hardware–model co-design, and that Pareto-based architecture modeling provides a practical alternative to exhaustive neural architecture search for on-device LLMs.

1 Introduction

Recent embodied AI systems increasingly incorporate large language models (LLMs) as high-level reasoning modules for semantic planning, long-horizon decision making, memory integration, and natural interaction [23, 49, 34]. In modern Vision–Language–Action (VLA) systems, LLMs transform multimodal observations into structured reasoning and action plans. However, deploying such capabilities on edge platforms such as autonomous vehicles and mobile robots remains challenging due to strict constraints on memory capacity, bandwidth, power consumption, and real-time latency [48, 46]. Architectures optimized for cloud-scale accelerators frequently become inefficient or infeasible in on-device settings: models with strong perplexity scaling may violate latency budgets,

*Luoyang Sun and Jiwen Jiang participated in a collaborative internship project between AI Lab, The Yangtze River Delta, and Li Auto, during which this work was conducted. Their affiliations with these two organizations reflect their participation in this project.

†Project leaders: Lei Ren and Chen Wei.

‡Corresponding authors: Haifeng Zhang and Cheng Deng. Email: `davendw49@gmail.com` (Cheng Deng).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Table 1: Representative Pareto trade-offs in LLM deployment systems. This work focuses on the model loss vs. inference latency trade-off in hardware-constrained on-device LLMs.

Scenario	Trade-off Targets	Key Optimization Factors
Memory allocation optimization	Throughput vs. single-query latency	Batch size selection and KV-cache reuse strategies
Quantization strategy selection	Memory cost vs. model precision	INT8 / INT4 quantization and accuracy calibration
Distributed inference systems	Communication cost vs. compute parallelism	Parallelism granularity and pipeline scheduling
Hardware-constrained on-device LLMs	**Model loss vs. inference latency**	**Model scaling and hardware constraints**

while aggressively optimized inference pipelines often incur substantial accuracy degradation. This tension motivates hardware–software co-design, where architectural decisions are explicitly guided by deployment constraints and hardware characteristics [17].

The challenge fundamentally stems from the irregular compute–memory behavior of transformer architectures [30, 12]. Self-attention is typically memory-bandwidth bound, feedforward layers are compute bound, and KV-cache management stresses on-chip memory capacity during autoregressive decoding. As a result, end-to-end inference performance is governed not only by model size, but also by arithmetic intensity, memory locality, KV-cache footprint, and MoE routing behavior. Small architectural modifications can shift kernels across regimes in the Roofline performance model [42, 19], substantially altering achievable hardware utilization. For example, scaling model depth and width increases memory I/O and reduces arithmetic intensity, often pushing inference into bandwidth-limited regimes on edge accelerators [5, 6].

Existing approaches to hardware-aware model design typically rely on neural architecture search (NAS), empirical benchmarking, or learned latency predictors [50, 14]. While effective in constrained settings, these methods often require training or profiling large numbers of candidate architectures, making adaptation across hardware platforms computationally expensive. Moreover, most existing approaches optimize latency or accuracy independently, without explicitly modeling the Pareto structure governing deployment trade-offs between model quality and inference efficiency. As shown in Table 1, LLM inference system design involves multiple competing objectives [9, 45, 25, 13, 47, 43, 12]. In this work, we focus on the interplay between model loss and inference latency in hardware-constrained on-device LLM deployment.

To address this challenge, we present PLAS (Pareto-optimal LLM Architecture Search), a hardware-aware framework for identifying Pareto-optimal on-device LLM architectures under deployment constraints. PLAS jointly models training loss and inference latency as functions of architectural design choices. Specifically, we fit an empirical architecture-to-loss model from trained LLMs and estimate inference latency using roofline-based hardware modeling. Combining these components enables efficient exploration of the accuracy–latency Pareto frontier without exhaustively training or profiling every candidate architecture.

We profile the inference latency of 1,942 candidate architectures on NVIDIA Jetson AGX Orin 64GB. For loss-model calibration, we train 170 representative models for 10B tokens each to characterize architecture-level scaling behavior under deployment constraints. At the same measured latency as Qwen2.5-0.5B on the target hardware, our co-designed architecture achieves 19.42% lower WikiText-2 perplexity. We further show that the framework generalizes across hardware platforms and workload configurations through lightweight hardware reparameterization. Our main contributions are summarized as follows:

- We present PLAS, a hardware-aware framework that jointly models architecture-level loss and inference latency, enabling explicit Pareto characterization of accuracy–latency trade-offs for on-device LLM deployment.
- We benchmark 1,942 LLM architectures on NVIDIA Jetson AGX Orin 64GB and train 170 representative models for 10B tokens each to characterize architecture-level scaling behavior and validate the architecture-to-loss model.
- We analyze the architectural trends emerging along the Pareto frontier, identifying consistent hardware-aligned design patterns including sparse MoE dominance, wide-and-shallow geometries, and phase-dependent expert configurations.
- We demonstrate that the proposed framework generalizes across hardware platforms and workload configurations through lightweight roofline reparameterization, enabling efficient deployment-oriented architecture exploration without exhaustive retraining.

2 Related Work

2.1 Efficient Architectures for On-Device LLMs

Recent LLM architectures increasingly prioritize inference efficiency beyond parameter scaling. Key directions include: (1) sparse activation via MoE routing [18, 2], (2) KV-cache reduction through MLA [12, 28] or sliding-window attention [40], (3) sub-quadratic attention mechanisms [41], and (4) hybrid architectures combining SSMs with attention [7]. These approaches improve inference efficiency through architectural innovation, but generally do not explicitly characterize deployment trade-offs between model quality and inference latency under hardware constraints.

2.2 Hardware-Aware LLM Deployment

On-device LLM deployment requires co-optimization across model, compression, and system layers. Compact models such as Qwen [44], MiniCPM [21], and SmolLM [4], together with quantization methods [27, 16] and inference systems such as vLLM [26], MLC-LLM [31], and PowerInfer [37], enable efficient deployment on heterogeneous hardware. Profiling tools such as LLM-Viewer [19] and LLMCompass [33] further provide hardware-aware analysis of transformer inference behavior. However, these works primarily optimize runtime execution rather than jointly modeling architecture-level accuracy and latency.

2.3 Neural Architecture Search and Hardware-Aware Optimization

Neural Architecture Search (NAS) [51, 29] automates architecture design through large-scale candidate exploration. Hardware-aware NAS methods [8] further incorporate latency or memory constraints into the search objective. HardCoRe-NAS [32] and ConNAS [11] address resource constraints in differentiable architecture search. HW-GPT-Bench [38] provides hardware-aware GPT architecture surrogates and supports multi-objective optimization. PLAS instead couples an architecture-to-loss model fitted under a fixed training recipe with operator-level roofline analysis, including sparse MoE designs. Existing studies on transformer scaling and depth–width trade-offs [39, 5, 6] show that architectural choices strongly affect inference efficiency, but are commonly studied under cloud-scale settings rather than deployment-oriented edge constraints.

Positioning. Prior work on transformer scaling, hardware-aware optimization, and neural architecture exploration [39, 5, 6] has shown that architectural choices strongly influence both model quality and inference efficiency. Rather than scaling models under unconstrained compute regimes, PLAS studies how LLM architectures "dance in fetters" under strict deployment constraints by jointly modeling validation loss and hardware-aware latency through empirical scaling behavior and roofline analysis. This enables efficient exploration of Pareto-optimal on-device LLM architectures without exhaustive candidate training or profiling.

3 PLAS: Hardware-constrained Pareto-optimal LLM Architecture Search

Classical scaling laws [22, 20] characterize the relationship between model size, data size, and training compute under fixed training budgets. In contrast, this work focuses on deployment, where the objective is to identify Pareto-optimal architectures under fixed hardware and inference latency constraints. In this section, we introduce PLAS, a deployment-oriented framework that jointly models architecture-level loss and inference latency for hardware-constrained on-device LLMs.

PLAS combines three components for deployment-oriented architecture exploration: architecture-to-loss modeling using empirical scaling behavior, roofline-based inference latency modeling under deployment workloads, and joint optimization to construct the Pareto frontier of hardware-constrained on-device LLM architectures.

3.1 Problem Formulation

Optimization Objective. We seek to minimize validation loss under latency and memory constraints:

$$\min_{\boldsymbol{\theta}\in\Theta} \ \mathcal{L}(\boldsymbol{\theta}) \ \text{ s.t. } \ T(\boldsymbol{\theta}; H, W) \le T_{\text{lat}}, \ \ M(\boldsymbol{\theta}; W) \le M_{\text{budget}} \tag{1}$$

where $\boldsymbol{\theta} = (l, d, d_m, r, \rho)$ denotes the architecture: l is the number of layers (depth), d the hidden dimension (width), $d_m = d_h \times n_{\text{kv}}$ the key–value cache dimension governing KV-cache memory footprint during decoding, r the total activated FFN expansion ratio ($r = Kd_{\text{ffn}}/d$, where d_{ffn} is the per-expert intermediate dimension and $K = 1$ for dense models), and $\rho = K/E \in (0, 1]$ the expert activation rate in mixture-of-experts (MoE) models, where K out of E experts are selected per token; for dense models $\rho = 1$. Here Θ denotes the configuration space. The five quantities above are the loss-model features; latency and memory evaluation additionally retain the complete configuration, including E, K, and attention-head counts. Detailed parameterization is provided in Appendix A.

The latency model $T(\boldsymbol{\theta}; H, W)$ models the end-to-end latency under context encoding and autoregressive generation, depending on both architectural and hardware characteristics. Hardware parameters $H = (\pi_H, \beta_H)$ include the peak compute throughput π_H (FLOPS) and memory bandwidth β_H (Bytes/s). Under roofline analysis, π_H and β_H jointly determine whether inference is compute- or bandwidth-bound. The workload configuration $W = (B, S_{\text{in}}, S_{\text{out}})$ specifies the batch size B, input sequence length S_{in}, and modeled decode-pass count S_{out}, which collectively affect attention complexity and KV-cache access patterns during decoding. The memory model $M(\boldsymbol{\theta}; W)$ estimates memory consumption based on model architecture and workload configuration.

Problem Tractability. Due to the high dimensionality of architectural hyperparameters in modern LLMs, the search space induced by Equation 1 is prohibitively large to enumerate. As a result, directly solving this constrained optimization problem is computationally infeasible in realistic deployment settings. Rather than performing brute-force search, we approximate the optimization landscape through explicit surrogate models that capture stable trends in both learning dynamics and system behavior. Specifically, in the following two subsections, we model validation loss using a empirical architecture-to-loss model fitted from empirical training jobs, and characterize inference latency via a roofline-based hardware performance model. This approach enables principled and scalable identification of Pareto-optimal architectures under fixed hardware constraints.

3.2 Architecture-to-Loss Modeling

Beyond reducing the cost of hyperparameter search, our goal is to capture systematic and generalizable relationships between architectural design choices and model quality. We therefore construct an explicit analytical surrogate for validation loss, approximating the true objective in Equation 1 using empirical scaling behavior observed during training.

As described in Section 4, we train 170 architectures covering both dense and MoE models and fit an empirical scaling law based on Equation 2. This parametric power-law approximation predicts validation loss under a fixed training recipe and token budget, facilitating efficient exploration of the target architecture space. Predictions outside the calibration range require additional validation.

Consistent with prior unified scaling analyses [10, 24], we model loss as a separable function of architecture components. Our empirical results reveal that sparsity-driven and base-capacity terms follow different width scaling exponents,

$$\hat{\mathcal{L}}(\boldsymbol{\theta}) = \frac{\kappa_l}{l^{\alpha_l}} + \frac{\kappa_\rho \cdot \rho^{\alpha_\rho}}{r^{\alpha_r} d^{\beta_1}} + \frac{\kappa_d}{r^{\alpha_r} d^{\beta_2}} + \frac{\kappa_m}{d_m^{\alpha_m}} + \mathcal{L}_\infty, \tag{2}$$

where $\phi = \{\kappa_l, \kappa_\rho, \kappa_d, \kappa_m, \alpha_l, \alpha_\rho, \alpha_r, \alpha_m, \beta_1, \beta_2, \mathcal{L}_\infty\}$ are fitted parameters.

3.3 Roofline-Based Latency Modeling

We model inference latency using roofline analysis [42], where latency is determined by the interaction between model compute, memory traffic, and hardware characteristics. Given peak compute throughput π_H (FLOPS) and memory bandwidth β_H (Bytes/s), the roofline model characterizes whether inference is compute- or bandwidth-bound through the arithmetic intensity $\mathcal{I} = \mathcal{F}/\mathcal{M}$ (FLOPs per byte). The resulting latency estimate is

$$\mathcal{T} = \max\left(\frac{\mathcal{F}}{\pi_H}, \frac{\mathcal{M}}{\beta_H}\right). \tag{3}$$

Here $\mathcal{F}$ and $\mathcal{M}$ denote FLOPs and transferred bytes for an operator, respectively; traffic $\mathcal{M}$ is distinct from resident memory M. We apply this estimate per operator and sum the operator latencies within each layer and stage. For a model with l layers and S_{in} input tokens, we model S_{out} decode

passes after prefill. Equation 4 gives the layer-wise latency approximation, with $S_{\text{in}} + t$ cached entries before decode pass $t = 0, \ldots, S_{\text{out}} - 1$. The numerical operator model also includes final normalization and the output projection. Completive FLOP counts and memory-traffic analysis are derived in Appendix H.

$$\hat{T_{\boldsymbol{\theta}}} = T_{\text{total}}(S_{\text{in}}, S_{\text{out}}) \simeq l \cdot T^{\text{pre}}_{\text{layer}}(S_{\text{in}}) + \sum_{t=0}^{S_{\text{out}}-1} l \cdot T^{\text{dec}}_{\text{layer}}(S_{\text{in}} + t), \tag{4}$$

3.4 Pareto Frontier Construction

During the experiments, we search for optimal LLM architectures via Pareto frontier analysis. An architecture $\boldsymbol{\theta}^\star$ is said to be Pareto-optimal if there exists no other architecture $\boldsymbol{\theta} \neq \boldsymbol{\theta}^\star$ such that

$$\nexists\, \boldsymbol{\theta} \neq \boldsymbol{\theta}^\star \quad \text{s.t. } \mathcal{L}(\boldsymbol{\theta}) \leq \mathcal{L}(\boldsymbol{\theta}^\star) \wedge\ T(\boldsymbol{\theta}; H, W) \leq T(\boldsymbol{\theta}^\star; H, W), \tag{5}$$

with at least one inequality strict.

The set of all such $\boldsymbol{\theta}^\star$ defines the Pareto frontier in the loss–latency plane. In the following section, we present empirical experiments for Pareto-optimal architecture discovery under fixed hardware and context constraints.

By combining the architecture-to-loss model with roofline-based latency analysis, PLAS jointly characterizes model quality and inference efficiency under deployment constraints. This formulation enables efficient exploration of Pareto-optimal on-device LLM architectures without exhaustive candidate training or profiling.

4 Empirical Pareto Frontier Analysis with PLAS

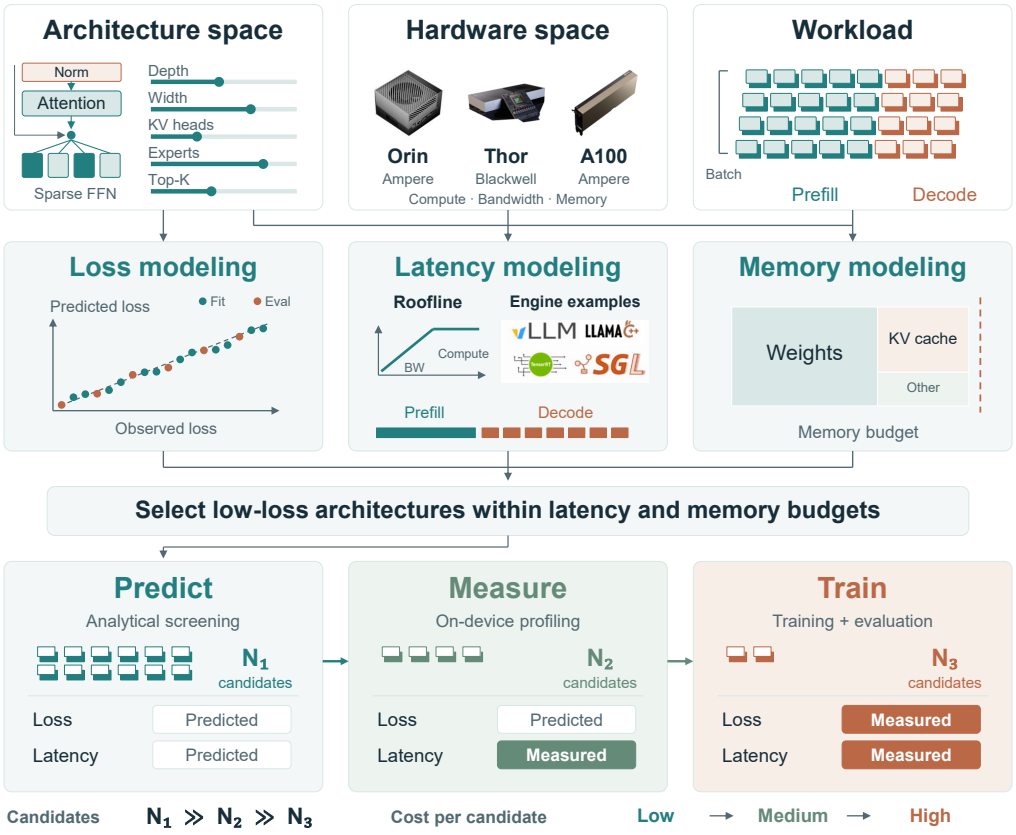
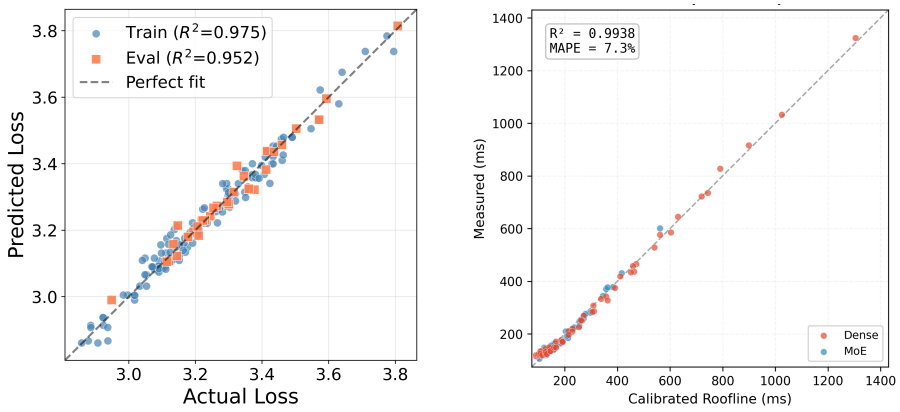


Figure 1: Overview of PLAS, a deployment-oriented framework for Pareto-optimal on-device LLM architecture exploration under hardware constraints.

This section presents **PLAS** (**P**areto-optimal **LLM** **A**rchitecture **S**earch), a framework that jointly models training loss and inference latency to enable hardware-aware architecture selection. We construct an empirical loss model from 170 trained architectures, characterize inference latency through roofline analysis, and integrate both to derive Pareto frontiers under deployment constraints. Figure 1 illustrates the overall workflow.

Pipeline. The 170 calibration models are used to fit and validate the loss surrogate before the search. Given the architecture-to-loss model (Section 3.2) and the roofline-based latency model (Section 3.3), PLAS performs deployment-oriented architecture exploration in three stages. Stage 1 analytically evaluates more than 10^5 candidate architectures and prunes the design space to a Pareto-optimal subset. Stage 2 deploys the surviving candidates on target hardware via vLLM and replaces analytical latency estimates with measured latency along the Pareto frontier. Stage 3 trains a small number of frontier architectures under the target data budget to validate the predicted loss. After loss-model calibration, this pipeline confines additional candidate training to a small selected set rather than the full search space. The study-wide count of 1,942 profiled architectures refers to latency measurements, whereas the Stage 1 count refers to analytical evaluations.

4.1 Architecture-to-Loss Model Validation



(a) Architecture scaling law validation.

(b) Calibrated roofline latency validation.

Figure 2: Validation of the two predictive components in PLAS. *Left*: architecture-to-loss scaling law achieving $R^2 = 0.975$ on 138 fitting architectures and $R^2 = 0.952$ on 32 held-out validation architectures. *Right*: calibrated roofline latency prediction on Jetson AGX Orin 64GB across 116 architectures, achieving $R^2 > 0.99$ for total latency estimation.

To obtain a predictive loss model, we train 170 Transformer configurations spanning depth, width, MoE sparsity, FFN expansion ratio, and KV-cache dimensions (detailed search space in Appendix A). Configurations are drawn by per-dimension stratified grids with granularity adapted to each dimension's loss sensitivity, supplemented with boundary configurations; the 170 calibration models are randomly partitioned into 138 fitting models and 32 held-out validation models. Each model is trained for 10B tokens under identical optimization settings (Appendix B). We fit a parametric scaling law of the form in Equation 2 using nonlinear least squares on the 138 fitting configurations.

As shown in Figure 2a, the fitted model achieves training $R^2 = 0.975$ and validation $R^2 = 0.952$ on held-out architectures spanning dense and sparse (MoE) configurations. These results support architecture-to-loss modeling under a fixed token budget and shared training recipe, consistent with prior scaling analyses [1, 24]. The fitted coefficients and functional form used for the analytical searches are provided in Appendix C.

4.2 Roofline Latency Validation

We estimate inference latency using roofline-based analysis, classifying operators as compute- or memory-bound according to arithmetic intensity and hardware capability. This enables joint prediction of loss and latency for **50,000+ architectures in 20 minutes**, without training or profiling those candidates. Separately, the study includes 1,942 architectures with measured inference latency.

Our primary analytical setting uses batch size 1, 1,024 input tokens, and 16 modeled decode passes. In the analytical plots, Decode denotes latency per pass, and Total is prefill latency plus 16 times the representative per-pass decode latency. Under this regime, prefill latency is dominated by attention computation, while decode latency is primarily constrained by weight loading. Different workloads therefore induce different optimization targets: decode latency for interactive applications, prefill latency for long-context processing, and total latency for balanced scenarios.

We validate the latency model on 116 architectures (60 Dense and 56 MoE) deployed on Jetson AGX Orin 64GB. As shown in Figure 2b, the calibrated roofline model achieves strong agreement with measured prefill, decode, and total latency, with $R^2 > 0.99$ and mean absolute percentage error below 10%. These error statistics characterize the calibrated Orin evaluation in this figure. These distinct latency regimes further lead to different Pareto-optimal architectural preferences under deployment constraints.

4.3 Pareto Frontier Analysis

Given explicit architecture-to-loss and latency models, we formulate architecture selection as a bi-objective optimization problem and identify Pareto-optimal architectures that jointly minimize validation loss and inference latency under hardware constraints.

4.3.1 Pareto Frontier Construction

Given loss predictions $\{\hat{\mathcal{L}}(\boldsymbol{\theta}_i)\}$ and latency estimates $\{\hat{T}(\boldsymbol{\theta}_i)\}$ for memory-feasible configurations, we identify the practical Pareto frontier as follows, requiring at least one of the two dominance inequalities to be strict:

$$\mathcal{P} = \{\boldsymbol{\theta}_i : \nexists\, \boldsymbol{\theta}_j \text{ s.t. } \hat{\mathcal{L}}(\boldsymbol{\theta}_j) \leq \hat{\mathcal{L}}(\boldsymbol{\theta}_i) \wedge \hat{T}(\boldsymbol{\theta}_j) \leq \hat{T}(\boldsymbol{\theta}_i)\} \tag{6}$$

We enumerate the specified finite grid, discard memory-infeasible configurations, and extract the predicted non-dominated set by sorting on latency and scanning loss. The search terminates after grid evaluation. For near-duplicate removal, points are grouped into latency bins within a factor of $1 + \epsilon$ of each bin's first point, retaining the lowest-loss point ($\epsilon = 0.02$). Plots may show a lower convex-hull subset of retained points rather than the complete non-dominated set.

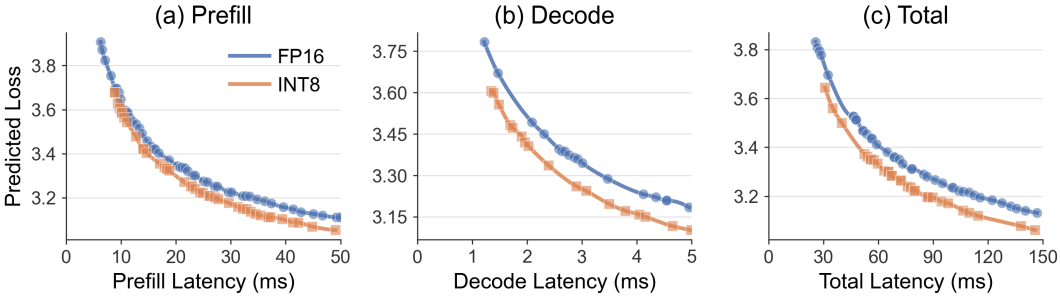


Figure 3: Predicted Pareto frontiers on NVIDIA Jetson AGX Orin 64GB, comparing FP16 and INT8: Prefill uses 1,024 input tokens; Decode is per-pass latency; Total is Prefill plus 16 times the representative Decode latency.

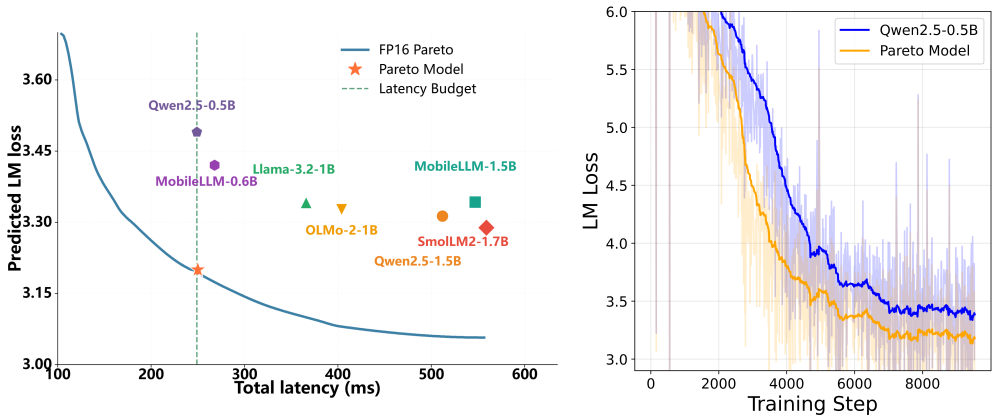
4.3.2 Precision and Workload Trade-offs

Figure 3 presents separate predicted frontiers for FP16 and INT8 under prefill, decode, and total latency objectives. Precision is fixed within each search. It affects weight and KV-cache storage, operator traffic, and latency. Equation 2 does not model quantization-induced quality loss. The predicted latency gains reflect the precision-dependent linear operators, non-linear operations, and precision-conversion overhead.

The Pareto frontier also varies substantially across deployment workloads. Prefill-oriented workloads favor architectures optimized for attention throughput under long contexts, whereas decode-oriented workloads favor architectures minimizing autoregressive memory access overhead. These observations indicate that optimal architecture selection depends strongly on workload characteristics in addition to hardware constraints.

4.3.3 Comparison with Existing On-Device LLMs

To situate the Pareto frontier against deployed on-device LLMs, we measure the inference latency of Qwen2.5-0.5B/1.5B, MobileLLM-0.6B/1.5B, Llama-3.2-1B, OLMo-2-1B, and SmolLM2-1.7B on the same Jetson AGX Orin and plot them against their scaling-law-predicted loss on the PLAS frontier (Figure 4a). Every published model lies above the frontier, indicating systematic accuracy headroom obtainable through hardware-aware architecture selection alone, without changes to the training protocol or data mix.



(a) Pareto search curves

(b) Training loss curves

Figure 4: (a) Predicted-loss vs. measured-latency comparison of existing on-device LLMs against the PLAS Pareto frontier model. The orange star marks the latency-matched co-designed architecture in the case study below. (b) Training curves of Qwen model and Pareto Frontier model.

Case Study. As a controlled instance, we use vLLM to identify a Pareto-optimal architecture matching the measured latency of Qwen2.5-0.5B on the target hardware (orange star in Figure 4a). The measured latencies are 250 ms for the co-designed model and 249 ms for Qwen2.5-0.5B under the same profiling setting. Trained using the same data mixture and optimization protocol, the co-designed architecture achieves **19.42%** lower WikiText-2 perplexity (50.88 vs. 63.14) at matched latency. This result demonstrates that substantial performance headroom remains available through hardware-aware architecture design alone, even under fixed training data and deployment constraints.

5 Model Architectural Evolution under Hardware Constraints

5.1 Architecture Evolution along the Pareto Frontier

The Pareto frontier enables deployment-oriented architecture selection under different latency budgets. Representative latency targets and corresponding application scenarios are summarized in Table 2.

Table 2: Latency requirements for representative applications.

Application	Latency	Rationale
Embodied AI	<20 ms (decode)	Real-time interaction
Smart Home	<500 ms (total)	Conversational
Autonomous Driving	<100 ms (total)	Safety-critical
Private Serving	<2 s (total)	Quality-focused

Figure 5 traces how Pareto-optimal architectures evolve as the latency budget increases, revealing several consistent patterns.

MoE Dominance. Sparse MoE architectures predominate among the displayed predicted frontier configurations under the evaluated grid and workloads. Under the batch-one constraint typical of on-device deployment, MoE models provide larger total capacity while maintaining comparable activated parameters per token, yielding superior loss–latency trade-offs relative to dense architectures.

Wide-and-Shallow Preference. In contrast to conventional deep-and-narrow transformer designs, the displayed predicted frontier shows a preference for wider and shallower geometries. Width increases

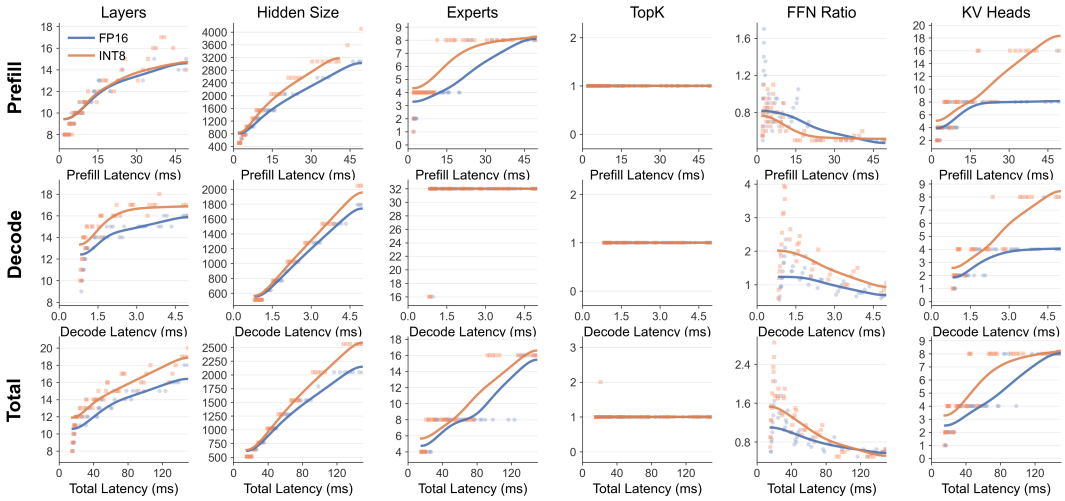
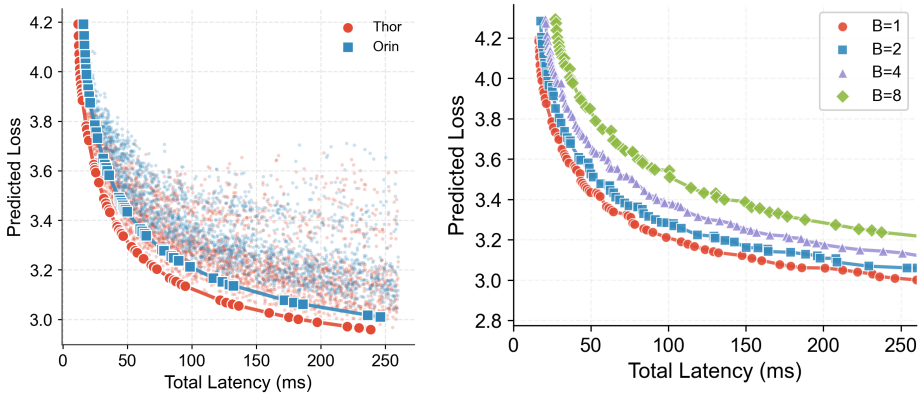


Figure 5: Architecture parameter evolution along Pareto frontiers under prefill- (top), per-pass decode- (middle), and total latency (bottom)-optimized objectives, comparing FP16 and INT8. Total uses prefill plus 16 representative decode passes. Connecting lines summarize discrete configurations.



(a) Cross-platform prediction (Total).

(b) Batch-size sensitivity (Total).

Figure 6: Analytical PLAS frontiers across hardware configurations and batch sizes. Total is prefill plus 16 times the representative per-pass decode latency. Full results are in Figure 7.

rapidly with the latency budget before saturating, after which additional capacity is allocated to depth, showing that width provides more efficient loss reduction per unit latency under hardware constraints.

Phase-Dependent Expert Configuration. The displayed frontiers show phase-dependent expert configurations. In prefill, distributing tokens across more FFN experts can reduce per-expert arithmetic intensity and increase weight traffic and execution overhead. In batch-one decode, increasing the expert pool while holding the activated experts and their sizes fixed increases capacity without directly increasing modeled active-weight traffic. Top-K=1 is frequent in the displayed Orin frontiers, but the selected routing configuration depends on hardware and workload.

Balanced Configuration under Total Latency. Total-latency optimization balances prefill and decode contributions. The selected expert count depends on the input–output workload, hardware, and enumerated configurations; it need not converge to a universal intermediate value.

5.2 Generalization across Hardware and Workload

PLAS transfers across hardware systems by updating the hardware inputs of the latency model and rerunning the analytical search; the loss model can be reused under the same training setup.

Cross-Platform Transfer. The loss model (Equation 2) is hardware-independent under a fixed training setup. We rerun Stage 1 with the Thor U analytical configuration and compare the predicted frontier with that for Jetson AGX Orin 64GB (Figure 6a). The Thor U analytical inputs are

350 TFLOP/s FP16 throughput, 273 GB/s bandwidth, and 64 GB memory. The plotted frontiers have similar shapes and different attainable trade-offs. This analytical comparison illustrates hardware reparameterization. The corresponding predicted architecture parameter evolution is reported in Appendix F.

Batch-Size Sensitivity. On-device deployment is dominated by single-stream inference (B=1), but small batches also occur in practice. Because PLAS is parameterized by the workload tuple $W = (B, S_{\text{in}}, S_{\text{out}})$, varying B requires no re-fitting of the loss model. Figure 6b overlays the Pareto frontiers for $B \in \{1, 2, 4, 8\}$: increasing B shifts each frontier upward in loss at matched latency, with the relative ordering of the prefill, decode, and total objectives preserved.

We summarize the observations within the evaluated grid and workloads as follows:

- Sparse MoE architectures dominate the Pareto frontier under on-device batch-one inference.
- The displayed frontier exhibits a wide-and-shallow preference rather than a uniform ordering at every latency budget.
- Prefill and decode phases favor substantially different expert configurations, making deployment workload a key factor in architecture selection.
- Under the precision-dependent latency model, the separately searched INT8 frontier shifts toward lower latency; quantization-induced quality changes are not modeled.
- Hardware-aware co-designed architectures consistently outperform existing on-device LLMs at matched latency and generalize across hardware platforms and workload configurations.

These observations provide actionable guidance for practitioners, while the full Pareto frontier enables precise architecture selection for specific deployment constraints.

Deployment across multiple devices. For a shared architecture, one can intersect relaxed Pareto bands from the target devices. Each retained candidate is within a chosen loss tolerance of each device's predicted frontier at that candidate's own latency. The intersection can be empty; this extension does not guarantee a universally optimal architecture. Appendix R gives the construction.

6 Conclusion and Limitations

We presented PLAS, a hardware-aware framework for Pareto-optimal on-device LLM architecture exploration under hardware constraints. By jointly modeling architecture-level loss and roofline-based inference latency, PLAS enables efficient characterization of the accuracy–latency Pareto frontier without exhaustive candidate training or profiling. Across 1,942 latency-profiled architectures, our results reveal consistent hardware-aligned design patterns and demonstrate that effective edge deployment requires explicit hardware–model co-design rather than heuristic architecture selection.

Limitations. The fitted loss model applies to the studied architecture family and 10B-token training recipe; changing either requires validation or refitting. Roofline estimates approximate runtime behavior, so deployment decisions still require measurements on the target device; the additional hardware searches are predictive analyses. Search results are conditional on the enumerated grid and do not establish global optimality. Precision-specific searches do not model quantization-induced quality loss. Extending PLAS to linear attention or state-space models requires corresponding loss and latency models. Appendix Q discusses these boundaries.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge AI Lab, The Yangtze River Delta, and Li Auto for providing the computational resources that supported this research. Their support enabled the model training and experimental evaluation reported in this paper. This work was carried out as part of a collaborative internship project between the two organizations, providing the setting for our study of hardware-aware language model design. We also thank the anonymous reviewers for their careful reading and constructive feedback, which helped improve the clarity and presentation of the paper.

References

[1] Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*, 2025.

[2] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b and gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

[3] 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.

[4] Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. `https://huggingface.co/blog/smollm3`, 2025.

[5] Song Bian, Minghao Yan, and Shivaram Venkataraman. Scaling inference-efficient language models. *arXiv preprint arXiv:2501.18107*, 2025.

[6] Song Bian, Tao Yu, Shivaram Venkataraman, and Youngsuk Park. Scaling laws meet model architecture: Toward inference-efficient llms. *arXiv preprint arXiv:2510.18245*, 2025.

[7] Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

[8] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware, 2019. URL `https://arxiv.org/abs/1812.00332`.

[9] Yihua Cheng, Yuhan Liu, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Kuntai Du, and Junchen Jiang. Lmcache: An efficient kv cache layer for enterprise-scale llm inference. *arXiv preprint arXiv:2510.09665*, 2025.

[10] Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pages 4057–4086. PMLR, 2022.

[11] Gregory De Ruyter, Mathias Verbeke, and Hans Hallez. Constraint-guided hardware-aware nas through gradient modification. In *The Fourteenth International Conference on Learning Representations*, 2026.

[12] Cheng Deng, Luoyang Sun, Jiwen Jiang, Yongcheng Zeng, Xinjian Wu, Wenxin Zhao, Qingfa Xiao, Jiachuan Wang, Haoyang Li, Lei Chen, et al. Plm: Efficient peripheral language models hardware-co-designed for ubiquitous computing. *arXiv preprint arXiv:2503.12167*, 2025.

[13] Vage Egiazarian, Roberto L Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, et al. Bridging the gap between promise and performance for microscaling fp4 quantization. *arXiv preprint arXiv:2509.23202*, 2025.

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[15] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022.

[16] 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.

[17] Cong Guo, Feng Cheng, Zhixu Du, James Kiessling, Jonathan Ku, Shiyu Li, Ziru Li, Mingyuan Ma, Tergel Molom-Ochir, Benjamin Morris, et al. A survey: Collaborative hardware and software design in the era of large language models. *IEEE Circuits and Systems Magazine*, 25 (1):35–57, 2025.

[18] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[19] Hahnyuan. Llm-viewer: Large language model visualization toolkit. `https://github.com/hahnyuan/LLM-Viewer`, 2024.

[20] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[21] 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.

[22] 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.

[23] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[24] Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.

[25] Eldar Kurtic, Alexandre Noll Marques, Shubhra Pandit, Mark Kurtz, and Dan Alistarh. "give me bf16 or give me death"? accuracy-performance trade-offs in llm quantization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26872–26886, 2025.

[26] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[27] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[28] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[29] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[30] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32431–32454. PMLR, 2024. URL `https://proceedings.mlr.press/v235/liu24ce.html`.

[31] MLC team. MLC-LLM, 2023-2025. URL https://github.com/mlc-ai/mlc-llm.

[32] Niv Nayman, Yonathan Aflalo, Asaf Noy, and Lihi Zelnik. Hardcore-nas: Hard constrained differentiable neural architecture search. In *International Conference on Machine Learning*, pages 7979–7990. PMLR, 2021.

[33] Princeton University. Llmcompass: Large language model performance profiling. https://github.com/PrincetonUniversity/LLMCompass, 2024.

[34] Ranjan Sapkota, Yang Cao, Konstantinos I Roumeliotis, and Manoj Karkee. Vision-language-action models: Concepts, progress, applications and challenges. *arXiv preprint arXiv:2505.04769*, 2025.

[35] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[36] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[37] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 590–606, 2024.

[38] Rhea S Sukthanker, Arber Zela, Benedikt Staffler, Aaron Klein, Lennart Purucker, Jörg K Franke, and Frank Hutter. Hw-gpt-bench: hardware-aware architecture benchmark for language models. *Advances in Neural Information Processing Systems*, 37:60776–60834, 2024.

[39] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. In *International Conference on Learning Representations*, 2022.

[40] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[41] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[42] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[43] Lang Xu, Kaushik Kandadi Suresh, Quentin Anthony, Nawras Alnaasan, and Dhabaleswar K Panda. Characterizing communication patterns in distributed large language model inference. In *2025 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 1–11. IEEE, 2025.

[44] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[45] Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. Kvlink: Accelerating large language models via efficient kv cache reuse. *arXiv preprint arXiv:2502.16002*, 2025.

[46] Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. Llm4drive: A survey of large language models for autonomous driving. *arXiv preprint arXiv:2311.01043*, 2023.

[47] Haofei Yin, Mengbai Xiao, Tinghong Li, Xiao Zhang, Dongxiao Yu, and Guanghui Zhang. Specpipe: Accelerating pipeline parallelism-based llm inference with speculative decoding, 2025. URL https://arxiv.org/abs/2504.04104.

[48] Yupeng Zhou, Can Cui, Juntong Peng, Zichong Yang, Juanwu Lu, Jitesh Panchal, Bin Yao, and Ziran Wang. A hierarchical test platform for vision language model (vlm)-integrated real-world autonomous driving. *ACM Transactions on Internet of Things*, 2025.

[49] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[50] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[51] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

A Architecture Search Space

We first define the architectural hyperparameters that constitute the architecture vector $\boldsymbol{\theta} = (l, d, d_m, r, \rho)$ introduced in Equation 1, then specify the calibration grid used in our scaling law study.

Depth and width. l denotes the number of transformer layers and d the hidden dimension. Together they determine the backbone geometry and dominate both parameter count and per-token compute.

KV-cache dimension. $d_m = d_h \times n_{\text{kv}}$ is the key–value cache dimension, where d_h is the per-head dimension and n_{kv} the number of key–value heads under grouped-query attention (GQA). Equivalently, $d_m = d/\text{gqa}$ with $\text{gqa} = n_h/n_{\text{kv}}$, where n_h is the number of query heads. This parameter directly governs KV-cache memory footprint and memory-bandwidth consumption during autoregressive decoding: each generated token appends $2d_m b_{\text{kv}}$ bytes to the cache, and all cached entries must be loaded at every subsequent decode step.

FFN expansion ratio. $r = Kd_{\text{ffn}}/d$ is the total activated FFN ratio, where d_{ffn} is the per-expert intermediate dimension; for dense models $K = 1$. For a gated architecture (SwiGLU), each FFN layer contributes $6rd^2$ FLOPs per token and $3rd^2 b_w$ bytes of weight traffic, making r the dominant factor in per-layer compute cost.

MoE activation rate. For mixture-of-experts models, $\rho = K/E \in (0, 1]$ denotes the fraction of experts activated per token, where K experts are routed from a pool of E. The expansion ratio r refers to the *total* effective ratio across all activated experts: if each expert has per-expert expansion r_{single}, then $r = K \cdot r_{\text{single}}$, ensuring that per-token FFN compute ($6rd^2$ FLOPs) is directly comparable between dense and sparse configurations. Crucially, ρ affects model *storage* (all E expert weights must reside in memory) but not per-token *computation* (only K experts are executed). For dense models, $E = K = 1$ and $\rho = 1$.

Table 3 specifies the architectural hyperparameters explored in our scaling law study.

Table 3: Architecture calibration space for scaling law fitting.

Parameter	Values
Depth l	$\{4, 8, 12, 16, 20, 24, 28, 32\}$
Width d	$\{768, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 3072\}$
MoE (E, K)	$\{(1, 1), (8, 1), (8, 2), (16, 1), (16, 2)\}$
GQA n_{kv}	$\{1, 2, 4, 8, n_h\}$

The calibration space jointly covers depth (4–32 layers), width (768–3072 hidden dimensions), MoE configurations (dense to 16 experts with Top-1/Top-2 routing), and grouped-query attention settings. This design ensures coverage of both dense and sparse architectures while avoiding degenerate or ill-conditioned regimes.

B Pre-training Details

All 170 model configurations are trained under identical conditions to ensure fair comparison:

Training Data. Each model is trained on 10B tokens from a mixture of general corpus, mathematics, and code data. This budget is sufficient to observe scaling behavior while remaining computationally tractable.

Optimization. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.01. The learning rate follows a cosine decay schedule from 1×10^{-4} to 1×10^{-6}, with linear warmup over the first 0.2% of training steps. QK-Norm is applied to stabilize MoE pre-training. All experiments use a global batch size of 256.

Evaluation. Model performance is measured by validation loss on a held-out set of approximately 1B tokens, averaged over the final 1,000 optimization steps to reduce variance. We additionally report perplexity on WikiText-2 for downstream evaluation.

Computational Cost. The calibration study comprises 170 models trained for 10B tokens each, or 1.7T model-tokens in aggregate. Using the approximate per-model allocation of eight GPUs for ten hours gives an estimate of $170 \times 8 \times 10 = 13{,}600$ GPU-hours. This estimate covers calibration training only. The fitted loss model can be reused across hardware targets and latency budgets under the same architecture family, training recipe, and token budget. The investment is most relevant to repeated architecture development; it may not be justified for choosing among only a few existing models.

The controlled case-study training curves are shown in Figure 4b.

C Scaling Law Coefficients

The fitted parametric loss model takes the following form:

$$\hat{\mathcal{L}}(\boldsymbol{\theta}) = \underbrace{\frac{\kappa_l}{l^{\alpha_l}}}_{\text{depth}} + \underbrace{\frac{\kappa_\rho \rho^{\alpha_\rho}}{r^{\alpha_r} d^{\beta_1}}}_{\text{sparsity-width}} + \underbrace{\frac{\kappa_d}{r^{\alpha_r} d^{\beta_2}}}_{\text{capacity}} + \underbrace{\frac{\kappa_m}{d_m^{\alpha_m}}}_{\text{KV-cache}} + \mathcal{L}_\infty \tag{7}$$

Table 4 reports the coefficients used to compute the analytical frontiers. The implementation uses $d/1024$ and $(256/d_m)^{\alpha_m}$. The coefficients below absorb these fixed reference scales: $\kappa_\rho = C\,1024^{\beta_1}$, $\kappa_d = D\,1024^{\beta_2}$, and $\kappa_m = F\,256^{\alpha_m}$, where C, D, F are the corresponding coefficients in the normalized implementation. Thus the functional form of Equation 2 is unchanged.

In the unnormalized convention of Equation 2, the concrete form is:

$$\begin{aligned}\hat{\mathcal{L}}(\boldsymbol{\theta}) = {} & \frac{10.000000}{l^{1.769567}} + \frac{0.001397888\,\rho^{1.246117}}{r^{0.100000}\,d^{-0.728499}} + \frac{79.914933}{r^{0.100000}\,d^{0.663562}} \\ & + \frac{1.022619}{d_m^{0.579612}} + 2.483950.\end{aligned} \tag{8}$$

Table 4: Fitted scaling law coefficients.

Term	Coefficient	Exponent	Interpretation
Depth	$\kappa_l = 10.000000$	$\alpha_l = 1.769567$	Depth contribution
Sparsity	$\kappa_\rho = 0.001397888$	$\alpha_\rho = 1.246117$, $\beta_1 = -0.728499$	Width-sparsity coupling
Capacity	$\kappa_d = 79.914933$	$\beta_2 = 0.663562$	Base capacity scaling
FFN ratio	–	$\alpha_r = 0.100000$	Unified FFN scaling
KV-cache	$\kappa_m = 1.022619$	$\alpha_m = 0.579612$	Cache efficiency
Irreducible		$\mathcal{L}_\infty = 2.483950$	

The negative width exponent in the sparsity term couples width and sparsity, while the capacity term decreases with width at fixed r. Relative sensitivity depends on the complete terms, not exponent magnitude alone.

D Latency Modeling Details

Roofline-Based Prediction. We analytically model latency using hardware roofline analysis. Each operator is classified as compute-bound or memory-bound based on its arithmetic intensity $I = \text{FLOPs}/\text{Bytes}$ relative to the hardware's compute-to-bandwidth ratio. Latency is estimated as:

$$T_{\text{op}} = \max\left(\frac{\text{FLOPs}}{\text{Peak Compute}}, \frac{\text{Memory Access}}{\text{Peak Bandwidth}}\right) \tag{9}$$

This approach enables rapid evaluation of over 50,000 configurations in minutes, making it suitable for large-scale architecture exploration.

Empirical Validation. We validate roofline predictions using vLLM with subprocess isolation to ensure accurate GPU memory accounting. Top Pareto candidates identified by analytical modeling are measured empirically to confirm their optimality.

Latency Scaling Behavior. Inference latency varies with workload parameters:

- **Sequence length:** Prefill latency scales quadratically with input length due to attention ($O(S^2)$); decode latency scales linearly due to KV-cache access ($O(S)$).
- **Output length:** Total latency is dominated by decode for long generations, making decode optimization critical for conversational applications.
- **Batch size:** Both phases benefit from batching with diminishing returns. Prefill becomes compute-bound at larger batches; decode remains memory-bound due to weight loading. For on-vehicle deployment, we focus on batch size 1.

Deployment Constraints. For a selected device, memory capacity is a feasibility constraint, while the latency budget specifies the application requirement. Precision changes the storage and traffic estimates; it does not imply an exact doubling of usable capacity because allocations need not all scale identically. Our evaluation concerns quality and latency on the given device. Board power and energy were not measured.

Timing Convention. The analytical searches use $T_{\text{total}} = T_{\text{pre}} + S_{\text{out}} T_{\text{dec,step}}$ with $S_{\text{out}} = 16$. The per-pass estimate uses a representative cached context of $S_{\text{in}} + \lfloor (S_{\text{out}} - 1)/2 \rfloor$. For ordinary autoregressive generation that samples the first output token from prefill, generating N tokens requires $N - 1$ subsequent decode passes. The batch-size sweeps use the operator-level numerical model.

Scenario-Specific Targets. The appropriate latency optimization target depends on deployment scenario:

- *Interactive/streaming applications* (chatbots, speculative decoding): optimize decode latency.
- *Long-context processing* (document QA): optimize prefill latency.
- *Balanced workloads* (summarization): optimize total latency.

E Full Generalization Results

Figure 7 presents the complete Pareto frontiers across all three latency metrics (Decode, Prefill, and Total). The top row compares the analytical configurations for Jetson AGX Orin 64GB and Thor U, showing that Thor U's frontier sits consistently below Orin's across all metrics, reflecting its higher compute throughput and memory bandwidth. The bottom row illustrates batch-size sensitivity on Orin for $B \in \{1, 2, 4, 8\}$, where larger batches generally shift the frontier toward higher loss at matched latency. This comparison concerns latency rather than throughput.

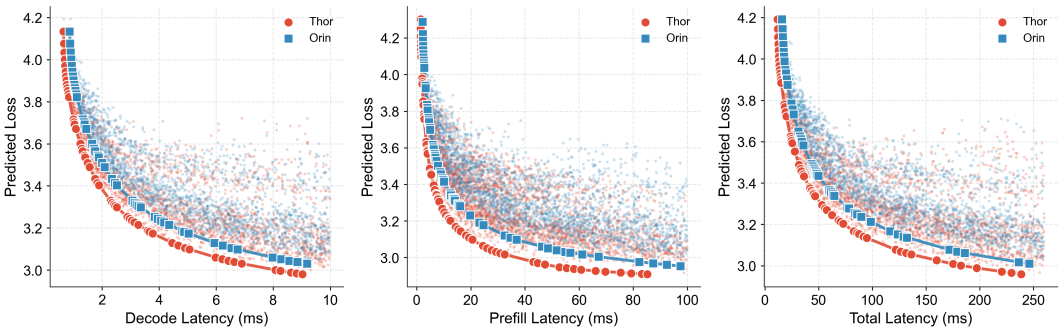
F Cross-Platform Architecture Evolution on Thor U

Figure 8 traces the predicted architecture parameters along the Thor U frontiers under prefill, decode, and total latency objectives. The selected expert counts, Top-K, depth, and width vary with the hardware configuration and latency budget; individual Orin selections, including Top-K=1, need not be preserved. The plots describe the sampled design space rather than a hardware-independent optimum.

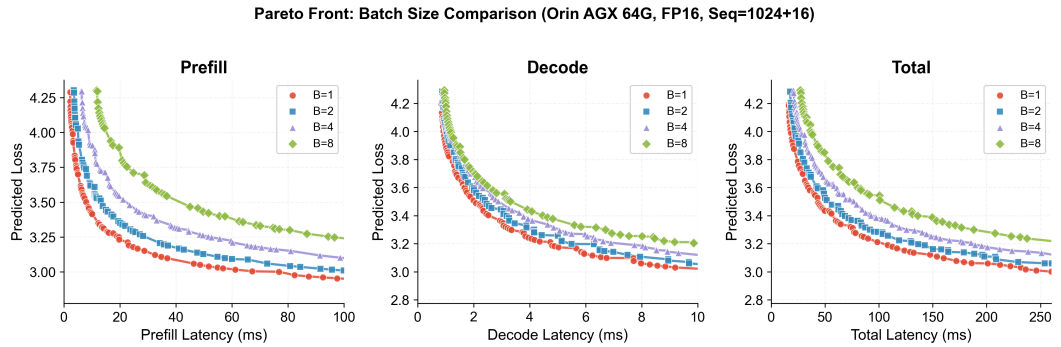
G Theoretical Framework for Hardware-Aware Architecture Optimization

G.1 From Empirical Search to Principled Optimization

Section 4.3 empirically discovered Pareto frontiers through large-scale search over 1,942 architectures. While effective, this approach raises fundamental questions: Can we predict optimal architectures



(a) Cross-platform: predicted Pareto frontiers for Jetson AGX Orin 64GB vs. the Thor U configuration (per-pass Decode / Prefill / Total). Total uses prefill plus 16 representative decode passes.



(b) Batch-size sensitivity on Orin: Pareto frontiers for $B \in \{1, 2, 4, 8\}$ across Prefill / per-pass Decode / Total. Total uses prefill plus 16 representative decode passes.

Figure 7: Full generalization results of the PLAS Pareto frontier across hardware platforms (top) and batch sizes (bottom). Abbreviated versions showing only the Total latency metric are in Figure 6.

without exhaustive search? What structural principles govern the Pareto frontier? How do solutions generalize to new hardware platforms?

This section addresses these questions by developing a theoretical framework that derives closed-form solutions for optimal architectures under different hardware constraint regimes. Rather than treating Pareto-optimal designs as empirical outcomes, we formalize architecture selection as an explicit constrained optimization problem.

Key Insight. Different hardware constraint regimes induce qualitatively distinct optimal solutions, particularly in how sparsity (MoE activation rate ρ) should be allocated. This explains why certain architectural patterns consistently emerge on the empirical Pareto frontier.

G.2 Problem Formulation and Constraint Types

We formalize the hardware co-design problem as:

$$\min_{\boldsymbol{\theta}} \quad \hat{L}(\boldsymbol{\theta}), \quad \text{subject to} \quad \hat{T}(\boldsymbol{\theta}; H, W) \leq T_{\text{lat}}, \quad \hat{M}(\boldsymbol{\theta}) \leq M_{\text{budget}} \tag{10}$$

where $\boldsymbol{\theta} = (l, d, r, \rho, \text{gqa})$ denotes depth, width, FFN ratio, activation rate, and GQA ratio.

Building on roofline analysis (Appendix H), we identify three constraint types. The **prefill constraint** (compute-bound) takes the form $l \cdot \xi_F \cdot d^2 \leq \bar{F}_p$ where $\xi_F = 4 + 4/\text{gqa} + 6r$. The **decode constraint** (bandwidth-bound) includes both weight loading and KV-cache access: $l \cdot (\xi_W^{\text{dec}} d^2 b_w + 2\bar{S} d b_{kv}/\text{gqa}) \leq \bar{M}_d$ where $\xi_W^{\text{dec}} = 2 + 2/\text{gqa} + 3r$. The **memory constraint** (storage-bound) approximates the dominant Transformer-block weight storage: $l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w \leq M_{\text{budget}}$ where $\xi_W^{\text{all}} = 2 + 2/\text{gqa} + 3r/\rho$. Here $\bar{S} = S_{\text{in}} + (S_{\text{out}} - 1)/2$ denotes the average context length during decoding. The storage expression omits embedding/output-head weights, KV-cache allocation, and runtime workspace.

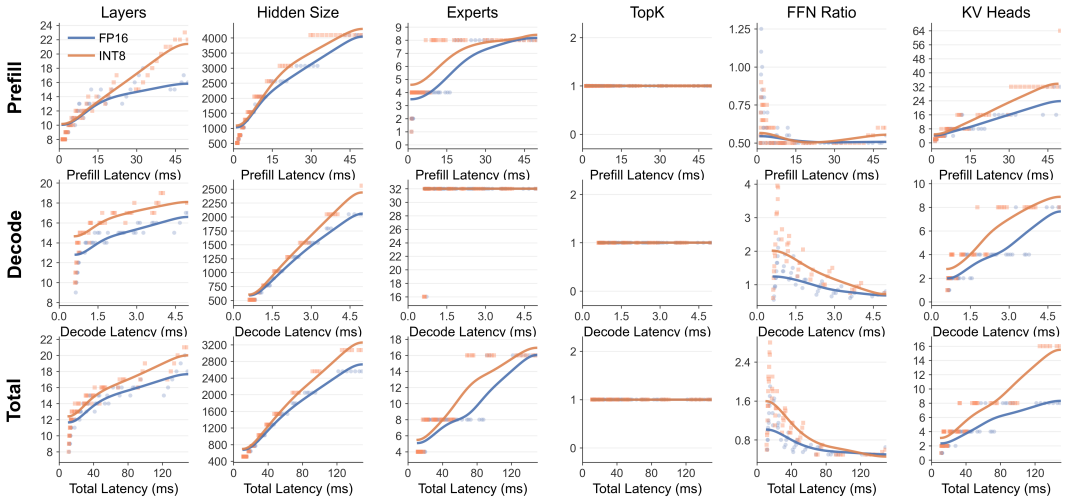


Figure 8: Architecture parameter evolution along the Pareto frontier for the Thor U analytical configuration, under prefill, decode, and total latency objectives, for FP16 and INT8 precision.

The numerical search filters candidates using estimated weight storage and KV-cache memory; deployment must additionally reserve runtime workspace.

Key Observation. The activation rate ρ appears only in ξ_W^{all}, reflecting that, at fixed effective FFN ratio r and other architectural variables, sparsity changes storage but not the modeled active computation. This asymmetry drives our main theoretical results.

G.3 Optimal Activation Rate Across Constraint Regimes

We characterize optimal activation rates ρ^* for three canonical regimes: latency-only (inference speed-limited with ample memory), memory-only (storage-limited with sufficient compute), and dual-constrained (tightly coupled hardware limits). These regimes naturally arise in different deployment scenarios: edge devices are often memory-constrained, automotive platforms are typically latency-constrained, and embedded systems are frequently dual-constrained.

Theorem G.1 (Latency-Constrained Regime)**.** *When only latency constraints are active (memory unconstrained):*

$$\rho^* = \rho_{\min} \tag{11}$$

Proof Sketch. Since $\partial \xi_F / \partial \rho = \partial \xi_W^{\text{dec}} / \partial \rho = 0$, the Lagrangian gradient $\partial \mathcal{L} / \partial \rho = \partial \hat{L} / \partial \rho > 0$ everywhere. Thus ρ^* occurs at the boundary. Full proof in Appendix I and Appendix L. □

Interpretation. At fixed effective FFN ratio r and other architectural variables, the simplified model favors reducing ρ within the feasible domain. One way to do this is to increase E while keeping K and each expert's width fixed: total storage increases, while modeled per-token active compute and weight traffic remain unchanged. This conditional result does not imply that changing K leaves latency unchanged or that top-1 routing is universally optimal. Memory capacity and implementation overhead constrain the benefit in deployment.

Theorem G.2 (Memory-Constrained Regime)**.** *For a continuous interior stationary point with only the memory constraint active:*

$$\rho^* = \left[\frac{\alpha_r \kappa_d}{(\alpha_\rho - \alpha_r)\kappa_\rho} \right]^{1/\alpha_\rho} \cdot d^{(\beta_1 - \beta_2)/\alpha_\rho} \tag{12}$$

valid when $\alpha_\rho > \alpha_r$ and $\rho_{\min} < \rho^* < 1$. Boundary and discrete candidates must be checked separately; this stationarity relation alone does not establish a global optimum.

Proof Sketch. Eliminating the Lagrange multiplier from KKT conditions for ρ and r yields an algebraic relation. Substituting the loss function and solving gives Equation 12. Complete derivation in Appendix J and Appendix M . □

Corollary G.3 (Width-Sparsity Scaling Law)**.** *For the interior memory-constrained relation, $\rho^* \propto d^{(\beta_1-\beta_2)/\alpha_\rho}$. The fitted exponents are $\beta_1 \approx -0.728499$, $\beta_2 \approx 0.663562$, and $\alpha_\rho \approx 1.246117$, giving an exponent of approximately -1.1171. Thus this relation associates greater width with lower activation rate.*

Interpretation. At fixed effective FFN ratio, lower ρ increases total expert storage. Using Appendix C, the continuous relation gives $\rho^*(1024) \approx 0.4024$ and $\rho^*(2048) \approx 0.1855$. These are conditional continuous values, not complete deployment configurations. Discrete (E, K) choices must satisfy the memory and latency constraints; for example, $K = 1, E = 8$ means $\rho = 0.125$.

Theorem G.4 (Dual-Constrained Regimes)**.** *When both constraints are active, their compatibility gives the following conditional activation rates for fixed remaining variables. The full stationary solution and its feasibility must still be checked.*

(a) Prefill + Memory:

$$\rho^* = \frac{3\eta_p b_w r}{\alpha_{attn}(2 - \eta_p b_w) + 6r}, \quad \eta_p = \bar{F}_p / M_{budget}, \quad 0 < \eta_p b_w \leq 2 \tag{13}$$

(b) Decode + Memory:

$$\rho^* = \frac{3\eta r}{\alpha_{attn}(1-\eta) + 3r + \delta}, \quad \xi_M^* = \frac{\alpha_{attn} + 3r + \delta}{\eta} \tag{14}$$

where $\eta = \bar{M}_d / M_{budget}$, $\alpha_{attn} = 2 + 2/gqa$, and $\delta = 2\bar{S} b_{kv}/(gqa \cdot d \cdot b_w)$ is the KV-cache correction. Here $\xi_M^* \triangleq \xi_W^{\text{all}}\big|_{\rho=\rho^*}$ denotes the equilibrium value of the storage coefficient (defined in Appendix H) under the dual constraint. For $r > 0$, $\eta > 0$, feasibility requires $\rho_{\min} \leq \rho^* \leq 1$; the upper bound is equivalent to $\eta \leq 1 + \delta/(\alpha_{attn} + 3r)$.

Proof Sketch. Constraint compatibility requires $\xi_F/(b_w \xi_W^{\text{all}}) = \eta_p$ (prefill case) or the decode analog. Solving these algebraic equations yields the stated forms. The KV-cache correction is additive in the numerator of the decode compatibility ratio. □

Comparison. Both cases admit a conditional closed form for the activation rate; the remaining variables remain coupled. The key difference is that the decode constraint includes a term proportional to $\bar{S}/\text{gqa}$ absent in the prefill case, creating stronger coupling between gqa and the latency budget. For systems with tight coupling between latency and memory such as automotive SoCs, practitioners should compute the constraint ratio η or η_p and apply the corresponding formula, verifying that $\rho_{\min} \leq \rho^* \leq 1$.

Table 5 provides a cross-reference between the theoretical results presented in this section and their complete derivations in the appendices.

Table 5: Cross-reference of theoretical results and detailed proofs.

Result	**Main Result**	**Detailed Proof**
Theorem G.1	$\rho^* = \rho_{\min}$	Appendix I (D1), Appendix L (P1)
Theorem G.2, Theorem G.3	$\rho^* \propto d^{(\beta_1-\beta_2)/\alpha_\rho}$	Appendix J (D2), Appendix M (P2)
Equation 13 in Theorem G.4	ρ^* prefill+memory	Appendix N (P3)
Equation 14 in Theorem G.4	ρ^* decode+memory	Appendix K (D3)
Equation 15–Equation 16	r^*, gqa^* closed forms	Appendix O

G.4 Optimal Depth, FFN Ratio, and GQA

The optimal depth always saturates the active constraint, taking the form $l^* = \bar{F}_p/(\xi_F d^2)$ for prefill, $l^* = \bar{M}_d/(\xi_W^{\text{eff}} d^2 b_w)$ for decode, or $l^* = M_{\text{budget}}/(\xi_W^{\text{all}} d^2 b_w)$ for memory constraints. The relation $l^* \propto d^{-2}$ applies when both the budget and the relevant coefficient are fixed. In particular, ξ_W^{eff} depends on d through KV-cache traffic, so the decode expression is not an exact inverse-square law in general. Along a frontier with varying budgets and configurations, depth and width may increase together.

The optimal FFN ratio and GQA ratio follow structured closed forms. We define the aggregate loss gradient $\tilde{D} \triangleq \kappa_\rho \rho^{\alpha_\rho} d^{\beta_2-\beta_1} + \kappa_d$, which combines the sparsity and base capacity contributions. Under prefill latency constraints, the solutions are:

$$r^* = \left[\frac{\alpha_r \tilde{D}}{6\alpha_l \kappa_l} \cdot \frac{\bar{F}_p^{\alpha_l}}{\xi_F^{\alpha_l - 1} d^{2\alpha_l + \beta_2}} \right]^{1/(\alpha_r+1)} \tag{15}$$

$$\mathrm{gqa}^* = \left[\frac{4\alpha_l \kappa_l}{\alpha_m \kappa_m} \cdot \frac{\xi_F^{\alpha_l - 1} d^{2\alpha_l + \alpha_m}}{\bar{F}_p^{\alpha_l}} \right]^{1/(\alpha_m+1)} \tag{16}$$

Other constraint regimes share the same structural form with modified coefficients and budget terms. Under decode latency constraints, the per-layer constraint coefficient generalizes to $\Gamma \triangleq \xi_W^{\mathrm{dec}} d^2 b_w + 2\bar{S} d b_{kv}/\mathrm{gqa}$, which accounts for KV-cache bandwidth; the r^* coefficient changes from $1/6$ to $1/3$ (since $\partial \xi_W^{\mathrm{dec}}/\partial r = 3$ vs. $\partial \xi_F/\partial r = 6$), and the gqa^* formula acquires an additional KV-cache coupling factor $(d^2 b_w + \bar{S} d b_{kv})$. Under memory constraints, an extra ρ^* factor enters the r^* numerator. Table 6 summarizes the key differences; complete formulas for all regimes are provided in Appendix O.

Table 6: Coefficient comparison across constraint regimes. C_r and C_g denote the constraint-derivative coefficients for r^* and gqa^*, respectively.

Regime	ρ^*	C_r	C_g	**Budget**	**Constraint Coeff.**
Prefill Latency	$\rho_{\min}$	6	4	$\bar{F}_p$	ξ_F
Decode Latency	$\rho_{\min}$	3	2	$\bar{M}_d$	Γ
Memory	Equation 12	$3/\rho$	2	M_{budget}	$\xi_W^{\mathrm{all}} d^2 b_w$
Prefill + Mem	Equation 13			see Appendix O	
Decode + Mem	Equation 14			see Appendix O	

The $2\times$ difference in coefficients between prefill and decode (e.g., $C_r = 6$ vs. 3) arises from the fundamental relation $\xi_F = 2\xi_W^{\mathrm{dec}}$, which reflects the FLOPs-to-memory-access ratio: each multiply-accumulate operation counts as 2 FLOPs but requires loading each weight parameter only once. Note that the decode $C_g = 2$ in Table 6 reflects only the weight-loading contribution; the full decode GQA derivative additionally includes a KV-cache correction term $2\bar{S} b_{kv}/(d \cdot b_w)$ (see Appendix I for the complete expression). These derivative coefficients alone do not determine the ordering of optimal FFN or GQA ratios: budgets, coupled variables, and KV-cache terms also enter the solutions.

H Problem Formulation and Roofline Analysis

This section presents a comprehensive roofline analysis for decoder-only Transformers, deriving FLOPs, memory traffic, and latency models for both prefill and decode phases.

H.1 Notation

H.2 Transformer Architecture

We consider a decoder-only Transformer with the following components per layer:

Multi-Head Attention. The attention mechanism uses n_h query heads and n_{kv} key-value heads (Grouped Query Attention). The hidden dimension is d, and the GQA ratio is $\mathrm{gqa} = n_h/n_{kv}$. The projection matrices are:

- Query projection $W_Q \in \mathbb{R}^{d \times d}$
- Key projection $W_K \in \mathbb{R}^{d \times d/\mathrm{gqa}}$
- Value projection $W_V \in \mathbb{R}^{d \times d/\mathrm{gqa}}$
- Output projection $W_O \in \mathbb{R}^{d \times d}$

Table 7: Symbol definitions for roofline analysis.

Symbol	Definition	Symbol	Definition
B	Batch size	S	Sequence length
l	Number of layers	d	Hidden dimension
d_h	Head dimension	n_h	Number of query heads
n_{kv}	Number of KV heads	gqa	GQA ratio $= n_h/n_{kv}$
d_m	KV dimension $= d/\text{gqa}$	r	FFN expansion ratio
E	Total experts	K	Active experts per token
ρ	Activation rate $= K/E$	b_w	Bytes per weight
b_a	Bytes per activation	b_{kv}	Bytes per KV element
π_H	Peak compute (FLOP/s)	β_H	Memory bandwidth (Bytes/s)
S_{in}	Input sequence length	S_{out}	Modeled decode-pass count

Feed-Forward Network. We consider a gated FFN (SwiGLU) with expansion ratio $r = d_{\text{ffn}}/d$:

- Up projection $W_{\text{up}} \in \mathbb{R}^{d \times rd}$
- Gate projection $W_{\text{gate}} \in \mathbb{R}^{d \times rd}$
- Down projection $W_{\text{down}} \in \mathbb{R}^{rd \times d}$

Mixture of Experts. In an MoE layer, the FFN is replicated into E experts, of which K are activated per token. The activation rate is $\rho = K/E$.

H.3 Per-Operator Analysis

H.3.1 Attention Projection Layers

Multi-head attention uses four linear projections: Query (Q), Key (K), Value (V), and Output (O). Modern architectures employ Grouped-Query Attention (GQA) [3] to reduce K/V dimensions by a factor of gqa.

Q Projection. Transform input $X \in \mathbb{R}^{B \times S \times d}$ to queries $Q \in \mathbb{R}^{B \times S \times d}$ via weight $W_Q \in \mathbb{R}^{d \times d}$.

Table 8: Q projection costs in prefill and decode phases.

Metric	Prefill	Decode ($S_q = 1$)
FLOPs	$2BSd^2$	$2Bd^2$
Weight Load	$d^2 \cdot b_w$	$d^2 \cdot b_w$
Activation Load	$BSd \cdot b_a$	$Bd \cdot b_a$
Activation Store	$BSd \cdot b_a$	$Bd \cdot b_a$

Arithmetic Intensity. From Table 8:

- **Prefill**: $\mathcal{I} \approx \frac{2BS}{b_w}$, typically compute-bound for large BS.
- **Decode**: $\mathcal{I} \approx \frac{2B}{b_w}$, typically memory-bound for small B.

K and V Projections (GQA). Project to reduced dimension $d_m = d/\text{gqa}$: $X \in \mathbb{R}^{B \times S \times d} \to K, V \in \mathbb{R}^{B \times S \times d_m}$ via $W_K, W_V \in \mathbb{R}^{d \times d_m}$.

GQA reduces K/V computation, weight memory, and critically KV-cache storage by factor gqa compared to Q/O projections.

O Projection. Identical to Q projection (see Table 8).

Table 9: K and V projection costs with GQA. Each operates on dimension $d_m = d/\text{gqa}$.

Metric (each)	Prefill	Decode
FLOPs	$2BSd^2/\text{gqa}$	$2Bd^2/\text{gqa}$
Weight Load	$d^2 b_w/\text{gqa}$	$d^2 b_w/\text{gqa}$
KV Cache Store	$BSd \cdot b_{kv}/\text{gqa}$	$Bd \cdot b_{kv}/\text{gqa}$

Summary. Total costs for Q, K, V, O projections:

$$\text{FLOPs} = 2BSd^2 \left(2 + \frac{2}{\text{gqa}}\right) \tag{17}$$

$$\text{Weight Memory} = d^2 b_w \left(2 + \frac{2}{\text{gqa}}\right) \tag{18}$$

For $\text{gqa} = 1$ (standard MHA): $8BSd^2$ FLOPs, $4d^2 b_w$ memory. For $\text{gqa} = 8$: $\sim$44% reduction.

H.3.2 Attention Score Computation

The attention mechanism computes scores through three main operations: query-key multiplication, softmax normalization, and score-value multiplication. We analyze each operation separately.

QK Matmul (Query-Key Attention Scores). For each attention head, compute attention scores by multiplying query $Q_h \in \mathbb{R}^{S_q \times d_h}$ with key transpose $K_h^T \in \mathbb{R}^{d_h \times S_{kv}}$, producing scores $\in \mathbb{R}^{S_q \times S_{kv}}$.

Table 10: Query-key attention score computation costs. Note $d_m = d/\text{gqa}$ is the reduced dimension for K cache with grouped-query attention.

Metric	Prefill ($S_q = S_{kv} = S$)	Decode ($S_q = 1$)
FLOPs	$2BS^2 d$	$2BSd$
Load Q	$BSd \cdot b_a$	$Bd \cdot b_a$
Load K Cache	$BSd_m \cdot b_{kv}$	$BSd_m \cdot b_{kv}$
Store Scores	$Bn_h S^2 \cdot b_a$	$Bn_h S \cdot b_a$

Prefill computes a full $S \times S$ attention matrix per head ($O(S^2)$ complexity), while decode only computes scores for one new token against S cached keys ($O(S)$ complexity).

Softmax Normalization. Attention scores are normalized using softmax, requiring approximately 5 operations per element: finding maximum (for numerical stability), subtraction, exponentiation, summation, and division.

Table 11: Softmax normalization costs over B batches, n_h heads, each processing $S_q \times S_{kv}$ scores.

Metric	Prefill ($S_q = S$)	Decode ($S_q = 1$)
FLOPs	$\approx 5Bn_h S^2$	$\approx 5Bn_h S$
Memory Load	$Bn_h S^2 \cdot b_a$	$Bn_h S \cdot b_a$
Memory Store	$Bn_h S^2 \cdot b_a$	$Bn_h S \cdot b_a$

Softmax FLOPs are typically negligible compared to matrix multiplications, contributing $O(n_h S^2)$ versus $O(Sd^2)$ for QK matmul when $d \gg n_h$.

Score-Value Matmul (Weighted Aggregation). Normalized attention scores $\in \mathbb{R}^{S_q \times S_{kv}}$ multiply value matrix $V_h \in \mathbb{R}^{S_{kv} \times d_h}$ to produce attention output $\in \mathbb{R}^{S_q \times d_h}$ per head.

Like QK matmul, score-value multiplication exhibits $O(S^2)$ complexity in prefill and $O(S)$ in decode.

Summary. Combining all three operations (QK matmul, softmax, score-value matmul):

Table 12: Score-value multiplication costs for weighted value aggregation.

Metric	**Prefill** ($S_q = S$)	**Decode** ($S_q = 1$)
FLOPs	$2BS^2d$	$2BSd$
Load Scores	$Bn_hS^2 \cdot b_a$	$Bn_hS \cdot b_a$
Load V Cache	$BSd_m \cdot b_{kv}$	$BSd_m \cdot b_{kv}$
Store Output	$BSd \cdot b_a$	$Bd \cdot b_a$

Table 13: Total computational and memory costs for attention score computation.

Metric	**Prefill** ($S_q = S$)	**Decode** ($S_q = 1$)
Total FLOPs	$4BS^2d + O(Bn_hS^2)$	$4BSd + O(Bn_hS)$
KV Cache Access	$2BSd_m \cdot b_{kv}$	$2BSd_m \cdot b_{kv}$

Key Observations. From Table 13:

- **Quadratic vs. Linear Scaling:** Prefill attention scales as $O(S^2)$ due to full $S \times S$ attention matrix, while decode scales linearly as $O(S)$ since only one new token attends to all previous tokens.
- **Projection Dominance:** For typical configurations where $d \gg S$ (e.g., $d = 4096$, $S = 2048$), projection FLOPs ($O(Bd^2)$) dominate over attention score FLOPs ($O(BSd)$), especially in decode phase.
- **KV Cache Bottleneck:** KV cache access remains $O(BSd_m)$ in both phases, becoming a critical bottleneck in decode when sequence length S is large.

H.3.3 FFN Layers

Modern transformers use either dense FFN layers or sparse Mixture-of-Experts (MoE) FFN layers.

Dense FFN with SwiGLU Activation. SwiGLU [35] uses three linear projections with gated activation:

1. **Gate projection**: $X \in \mathbb{R}^{B\times S\times d} \xrightarrow{W_g} G \in \mathbb{R}^{B\times S\times rd}$
2. **Up projection**: $X \in \mathbb{R}^{B\times S\times d} \xrightarrow{W_u} U \in \mathbb{R}^{B\times S\times rd}$
3. **Gated activation**: $H = \text{SiLU}(G) \odot U$ (element-wise)
4. **Down projection**: $H \in \mathbb{R}^{B\times S\times rd} \xrightarrow{W_d} Y \in \mathbb{R}^{B\times S\times d}$

where r is the expansion ratio (typically $r = \frac{8}{3}$ or $r = 4$).

Table 14: SwiGLU FFN costs. Element-wise ops contribute negligible $O(BSrd)$ FLOPs vs $O(BSrd^2)$ for projections.

Component	FLOPs (Prefill/Decode)	Weight Memory
Gate ($W_g \in \mathbb{R}^{d\times rd}$)	$2BSrd^2$ / $2Brd^2$	$rd^2 \cdot b_w$
Up ($W_u \in \mathbb{R}^{d\times rd}$)	$2BSrd^2$ / $2Brd^2$	$rd^2 \cdot b_w$
Down ($W_d \in \mathbb{R}^{rd\times d}$)	$2BSrd^2$ / $2Brd^2$	$rd^2 \cdot b_w$
Total (3 projections)	$6BSrd^2$ / $6Brd^2$	$3rd^2 \cdot b_w$

Mixture-of-Experts (MoE) FFN. MoE [36, 15] replaces dense FFN with E parallel experts, activating only top-K per token via learned routing, decoupling computation from capacity.

Key parameters: E = total experts; K = active experts/token (typically $K = 2$); $\rho = K/E$ = activation rate; r_{single} = per-expert expansion; $r = K \cdot r_{\text{single}}$ = effective expansion.

Costs: FLOPs match dense FFN with same effective r, but all E experts must be stored:

$$\text{MoE FLOPs} = 6BSrd^2 \text{ (prefill)}, \quad 6Brd^2 \text{ (decode)} \tag{19}$$

$$\text{MoE Weight Memory} = \frac{3rd^2 b_w}{\rho} = E \cdot 3r_{\text{single}} d^2 b_w \tag{20}$$

Table 15: Dense vs MoE FFN comparison. MoE achieves $1/\rho$ more capacity at same FLOPs.

Architecture	**FLOPs** Prefill	Decode	**Weight Memory** Total
Dense FFN	$6BSrd^2$	$6Brd^2$	$3rd^2 b_w$
MoE FFN	$6BSrd^2$	$6Brd^2$	$\frac{3rd^2 b_w}{\rho}$
Ratio (MoE/Dense)	$1\times$	$1\times$	E/K

Key Observations.

- **Capacity-Computation Decoupling:** MoE scales parameters without increasing FLOPs. Example: $E = 8$, $K = 2$ ($\rho = 0.25$) yields $4\times$ more parameters at same compute.
- **Storage and Decode Traffic:** At fixed effective ratio r, total FFN weight storage grows as $1/\rho$, whereas the batch-one model reads only the activated experts per decode pass. Total expert storage therefore does not directly multiply per-pass weight traffic by $1/\rho$. Quantization can reduce storage and traffic; actual latency also depends on routing, caching, and scheduling.
- **FFN Dominance:** FFN accounts for ∼60-70% of total FLOPs when $r \approx 4$ (vs attention projection costs in Equation 17 and Equation 18).

H.4 Per-Layer Coefficient Summary

Combining attention projections and FFN, the per-layer costs are:

$$\text{FLOPs per layer} = \underbrace{2BSd^2\left(2 + \frac{2}{\text{gqa}}\right)}_{\text{Attention Proj.}} + \underbrace{6BSrd^2}_{\text{FFN}} + \underbrace{O(BS^2 d)}_{\text{Attn Score}} \tag{21}$$

$$\text{Weight (compute)} = \underbrace{d^2 b_w\left(2 + \frac{2}{\text{gqa}}\right)}_{\text{Attention Proj.}} + \underbrace{3rd^2 b_w}_{\text{FFN (active)}} \tag{22}$$

$$\text{Weight (storage)} = \underbrace{d^2 b_w\left(2 + \frac{2}{\text{gqa}}\right)}_{\text{Attention Proj.}} + \underbrace{\frac{3rd^2 b_w}{\rho}}_{\text{FFN (all experts)}} \tag{23}$$

For large d where projection FLOPs dominate, we define the normalized coefficients:

$$\xi_F = 4 + \frac{4}{\text{gqa}} + 6r \quad \text{(FLOPs coefficient)} \tag{24}$$

$$\xi_W^{\text{dec}} = 2 + \frac{2}{\text{gqa}} + 3r \quad \text{(Decode weight coefficient)} \tag{25}$$

$$\xi_W^{\text{all}} = 2 + \frac{2}{\text{gqa}} + \frac{3r}{\rho} \quad \text{(Storage coefficient)} \tag{26}$$

Key Relations:

1. $\xi_F = 2 \cdot \xi_W^{\text{dec}}$ for any (r, gqa). This factor of 2 arises because each multiply-accumulate operation counts as 2 FLOPs but requires loading each weight only once.

2. ρ appears only in ξ_W^{all}, not in ξ_F or ξ_W^{dec}. This reflects that sparsity affects storage but not per-token computation.
3. For dense models ($\rho = 1$): $\xi_W^{\text{all}} = \xi_W^{\text{dec}}$.

H.5 Inference Phases and Roofline Model

Autoregressive generation consists of two phases:

Prefill Phase. The model processes S_{in} input tokens in parallel. This phase is typically **compute-bound**.

Decode Phase. The analytical workload consists of S_{out} decode passes after prefill. Before pass $t = 0, \ldots, S_{\text{out}} - 1$, the KV-cache contains $S_{\text{in}} + t$ entries. This phase is typically **memory-bandwidth-bound**.

Roofline Model. The roofline model [42] characterizes kernel latency as:

$$T = \max\left(\frac{\mathcal{F}}{\pi_H}, \frac{\mathcal{W}}{\beta_H}\right) \tag{27}$$

where $\mathcal{F}$ is FLOPs, $\mathcal{W}$ is memory traffic, π_H is peak compute, and β_H is memory bandwidth.

H.6 Latency Modeling

H.6.1 Prefill Latency

For batch size B and input sequence length S_{in}, the total FLOPs per layer is:

$$\mathcal{F}_{\text{layer}} = BS_{\text{in}}d^2 \cdot \xi_F \tag{28}$$

With large batch-sequence product BS_{in}, prefill is typically compute-bound:

$$T_{\text{pre}} = \frac{l \cdot BS_{\text{in}}d^2 \cdot \xi_F}{\pi_H} \tag{29}$$

H.6.2 Decode Latency: Single Step Analysis

The following closed-form decode approximation assumes $B = 1$ and retains activated-weight and existing KV-cache reads. It omits smaller activation and cache-write terms, which are included in the numerical operator model. At decode pass t, the cached context length is $S_{\text{in}} + t$:

Memory Traffic per Step. (1) Weight loading:

$$\mathcal{W}_{\text{weight}} = \xi_W^{\text{dec}} \cdot d^2 \cdot b_w \tag{30}$$

(2) KV-cache loading:

$$\mathcal{W}_{\text{KV}}(t) = \frac{2(S_{\text{in}} + t) \cdot d \cdot b_{kv}}{\text{gqa}} \tag{31}$$

Total per layer:

$$\mathcal{W}_{\text{layer}}(t) = \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2(S_{\text{in}} + t) \cdot d \cdot b_{kv}}{\text{gqa}} \tag{32}$$

Single Step Latency. Under this $B = 1$ approximation, decode is typically memory-bound:

$$T_{\text{step}}(t) = \frac{l}{\beta_H}\left[\xi_W^{\text{dec}} d^2 b_w + \frac{2(S_{\text{in}} + t)db_{kv}}{\text{gqa}}\right] \tag{33}$$

H.6.3 Decode Latency: Total Latency

Summing over $t = 0, \ldots, S_{\text{out}} - 1$:

$$T_{\text{decode}} = \frac{l}{\beta_H} \left[S_{\text{out}} \cdot \xi_W^{\text{dec}} d^2 b_w + \frac{2db_{kv}}{\text{gqa}} \sum_{t=0}^{S_{\text{out}}-1} (S_{\text{in}} + t) \right] \tag{34}$$

Evaluating the sum:

$$\sum_{t=0}^{S_{\text{out}}-1} (S_{\text{in}} + t) = S_{\text{out}} \cdot S_{\text{in}} + \frac{S_{\text{out}}(S_{\text{out}} - 1)}{2} = S_{\text{out}} \cdot \bar{S} \tag{35}$$

where the **average context length** is:

$$\bar{S} = S_{\text{in}} + \frac{S_{\text{out}} - 1}{2} \tag{36}$$

Complete Decode Latency.

$$T_{\text{decode}} = \frac{l \cdot S_{\text{out}}}{\beta_H} \left[\xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2\bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} \right] \tag{37}$$

H.6.4 Decode Latency: Unified Form

Define the **effective decode coefficient**:

$$\xi_W^{\text{eff}}(d, \text{gqa}) = \xi_W^{\text{dec}} + \frac{2\bar{S} \cdot b_{kv}}{\text{gqa} \cdot d \cdot b_w} \tag{38}$$

Then:

$$T_{\text{decode}} = \frac{l \cdot S_{\text{out}} \cdot \xi_W^{\text{eff}} \cdot d^2 \cdot b_w}{\beta_H} \tag{39}$$

Expanding ξ_W^{eff}:

$$\xi_W^{\text{eff}} = 2 + \frac{2}{\text{gqa}} + 3r + \frac{2\bar{S} \cdot b_{kv}}{\text{gqa} \cdot d \cdot b_w} \tag{40}$$

Note: ξ_W^{eff} depends on d and gqa, which introduces additional coupling in optimization problems.

H.7 Memory Footprint

Per-layer storage (all E experts):

$$M_{\text{layer}} = \xi_W^{\text{all}} \cdot d^2 \cdot b_w \tag{41}$$

Total model memory:

$$M = l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w \tag{42}$$

H.8 Constraint Formulations

H.8.1 Prefill Constraint

Given target latency $T_{\text{lat}}^{\text{pre}}$, define $\bar{F}_p = T_{\text{lat}}^{\text{pre}} \cdot \pi_H / (BS_{\text{in}})$:

$$l \cdot \xi_F \cdot d^2 \leq \bar{F}_p \tag{43}$$

H.8.2 Decode Constraint

Given target latency $T_{\text{lat}}^{\text{dec}}$, define $\bar{M}_d = T_{\text{lat}}^{\text{dec}} \cdot \beta_H / S_{\text{out}}$:

$$l \cdot \xi_W^{\text{eff}} \cdot d^2 \cdot b_w \leq \bar{M}_d \tag{44}$$

Expanding with the full KV-cache term:

$$l \cdot \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2l \cdot \bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} \leq \bar{M}_d \tag{45}$$

H.8.3 Memory Constraint

$$l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w \leq M_{\text{budget}} \tag{46}$$

H.9 Summary

Table 16: Roofline coefficients and their usage.

Coefficient	Formula	Usage
ξ_F	$4 + \frac{4}{\text{gqa}} + 6r$	Prefill FLOPs: $\mathcal{F} = l \cdot \xi_F \cdot BS \cdot d^2$
ξ_W^{dec}	$2 + \frac{2}{\text{gqa}} + 3r$	Decode weight traffic
ξ_W^{eff}	$\xi_W^{\text{dec}} + \frac{2\bar{S}b_{kv}}{\text{gqa} \cdot d \cdot b_w}$	Decode total traffic (weight + KV)
ξ_W^{all}	$2 + \frac{2}{\text{gqa}} + \frac{3r}{\rho}$	Storage: $M = l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w$

Table 17: Partial derivatives of coefficients.

	$\partial/\partial r$	$\partial/\partial \text{gqa}$	$\partial/\partial \rho$	$\partial/\partial d$
ξ_F	6	$-\frac{4}{\text{gqa}^2}$	0	0
ξ_W^{dec}	3	$-\frac{2}{\text{gqa}^2}$	0	0
ξ_W^{eff}	3	$-\frac{2}{\text{gqa}^2} - \frac{2\bar{S}b_{kv}}{\text{gqa}^2 d b_w}$	0	$-\frac{2\bar{S}b_{kv}}{\text{gqa}d^2 b_w}$
ξ_W^{all}	$\frac{3}{\rho}$	$-\frac{2}{\text{gqa}^2}$	$-\frac{3r}{\rho^2}$	0

Note: ξ_W^{eff} depends on d, which introduces additional coupling in the optimization. For dense models ($\rho = 1$), we have $\xi_W^{\text{all}} = \xi_W^{\text{dec}}$.

Additionally, we define the **aggregate loss gradient** $\tilde{D} \triangleq \kappa_\rho \rho^{\alpha_\rho} d^{\beta_2 - \beta_1} + \kappa_d$, which combines the sparsity and base capacity contributions from the loss function. This shorthand appears throughout the following case derivations (Cases D1–D3, P1–P3) in the stationarity conditions for r.

I Case D1: Decode, Latency-Constrained

I.1 Problem Statement

$$\begin{aligned} \min_{l,d,r,\text{gqa},\rho} \quad & \hat{\mathcal{L}}(\theta) = \frac{\kappa_l}{l^{\alpha_l}} + \frac{\kappa_\rho \rho^{\alpha_\rho}}{r^{\alpha_r} d^{\beta_1}} + \frac{\kappa_d}{r^{\alpha_r} d^{\beta_2}} + \frac{\kappa_m \cdot \text{gqa}^{\alpha_m}}{d^{\alpha_m}} + \hat{\mathcal{L}}_\infty \\ \text{s.t.} \quad & l \cdot \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2l \cdot \bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} = \bar{M}_d \\ & \rho \geq \rho_{\min} \end{aligned} \tag{47}$$

where $\xi_W^{\text{dec}} = 2 + 2/\text{gqa} + 3r$.

Define the constraint function:

$$g_T(\theta) = l \cdot \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2l \cdot \bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} - \bar{M}_d \tag{48}$$

I.2 Lagrangian

$$\mathscr{L} = \hat{\mathcal{L}}(\theta) + \mu_T \cdot g_T(\theta) \tag{49}$$

I.3 Constraint Partial Derivatives

$$\frac{\partial g_T}{\partial l} = \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2\bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} \tag{50}$$

$$\frac{\partial g_T}{\partial d} = 2l \cdot \xi_W^{\text{dec}} \cdot d \cdot b_w + \frac{2l \cdot \bar{S} \cdot b_{kv}}{\text{gqa}} \tag{51}$$

$$\frac{\partial g_T}{\partial r} = 3l \cdot d^2 \cdot b_w \tag{52}$$

$$\frac{\partial g_T}{\partial \text{gqa}} = -\frac{2l \cdot d^2 \cdot b_w}{\text{gqa}^2} - \frac{2l \cdot \bar{S} \cdot d \cdot b_{kv}}{\text{gqa}^2} \tag{53}$$

$$\frac{\partial g_T}{\partial \rho} = 0 \tag{54}$$

Note: $\partial g_T / \partial \rho = 0$ still holds because the KV-cache term is also independent of ρ.

I.4 KKT Conditions

Stationarity for l:

$$-\frac{\alpha_l \kappa_l}{l^{\alpha_l+1}} + \mu_T \left(\xi_W^{\text{dec}} d^2 b_w + \frac{2\bar{S} d b_{kv}}{\text{gqa}} \right) = 0 \tag{55}$$

Solving for μ_T:

$$\mu_T = \frac{\alpha_l \kappa_l}{l^{\alpha_l+1} \left(\xi_W^{\text{dec}} d^2 b_w + \frac{2\bar{S} d b_{kv}}{\text{gqa}} \right)} \tag{56}$$

Stationarity for ρ:

$$\frac{\partial \mathscr{L}}{\partial \rho} = \frac{\alpha_\rho \kappa_\rho \rho^{\alpha_\rho - 1}}{r^{\alpha_r} d^{\beta_1}} + \mu_T \cdot 0 = \frac{\alpha_\rho \kappa_\rho \rho^{\alpha_\rho - 1}}{r^{\alpha_r} d^{\beta_1}} > 0 \tag{57}$$

Since $\partial \mathscr{L} / \partial \rho > 0$ for all $\rho > 0$:

$$\rho^* = \rho_{\min} \tag{58}$$

Physical interpretation. The activation rate ρ does not appear in the decode latency constraint ($\partial g_T / \partial \rho = 0$), because only K experts are activated per token regardless of the total pool size E—thus per-token computation and bandwidth cost are invariant to ρ. Meanwhile, the loss is monotonically increasing in ρ: fewer activated experts relative to total experts means greater total model capacity at no additional per-token cost. Therefore, the optimal strategy under latency constraints is to maximize sparsity (minimize ρ), i.e., increase the expert pool E as far as memory permits while keeping K fixed.

Stationarity for r:

$$-\frac{\alpha_r \tilde{D}}{r^{\alpha_r+1} d^{\beta_2}} + 3\mu_T l d^2 b_w = 0 \tag{59}$$

Stationarity for gqa:

$$\frac{\alpha_m \kappa_m \cdot \text{gqa}^{\alpha_m - 1}}{d^{\alpha_m}} - \mu_T \left(\frac{2l d^2 b_w}{\text{gqa}^2} + \frac{2l \bar{S} d b_{kv}}{\text{gqa}^2} \right) = 0 \tag{60}$$

Simplifying:

$$\frac{\alpha_m \kappa_m \cdot \text{gqa}^{\alpha_m - 1}}{d^{\alpha_m}} = \frac{2\mu_T l}{\text{gqa}^2} \left(d^2 b_w + \bar{S} d b_{kv} \right) \tag{61}$$

Stationarity for d:

$$-\frac{\beta_1\kappa_\rho\rho^{\alpha_\rho}}{r^{\alpha_r}d^{\beta_1+1}}-\frac{\beta_2\kappa_d}{r^{\alpha_r}d^{\beta_2+1}}-\frac{\alpha_m\kappa_m\mathrm{gqa}^{\alpha_m}}{d^{\alpha_m+1}}+\mu_T\left(2l\xi_W^{\mathrm{dec}}db_w+\frac{2l\bar{S}b_{kv}}{\mathrm{gqa}}\right)=0 \tag{62}$$

I.5 Solution Derivation

Step 1: Depth. From the active constraint:

$$l^*=\frac{\bar{M}_d}{\xi_W^{\mathrm{dec}}d^2b_w+\frac{2\bar{S}db_{kv}}{\mathrm{gqa}}} \tag{63}$$

Or equivalently using ξ_W^{eff}:

$$l^*=\frac{\bar{M}_d}{\xi_W^{\mathrm{eff}}\cdot d^2\cdot b_w} \tag{64}$$

Step 2: Multiplier. Define the effective constraint coefficient:

$$\Gamma\triangleq\xi_W^{\mathrm{dec}}d^2b_w+\frac{2\bar{S}db_{kv}}{\mathrm{gqa}}=\xi_W^{\mathrm{eff}}\cdot d^2\cdot b_w \tag{65}$$

Then:

$$\mu_T l=\frac{\alpha_l\kappa_l}{l^{\alpha_l}\Gamma}=\frac{\alpha_l\kappa_l\Gamma^{\alpha_l-1}}{\bar{M}_d^{\alpha_l}} \tag{66}$$

Step 3: FFN Ratio. From stationarity for r:

$$\frac{\alpha_r\tilde{D}}{r^{\alpha_r+1}d^{\beta_2}}=3\mu_T ld^2b_w \tag{67}$$

$$r^*=\left[\frac{\alpha_r\tilde{D}}{3\alpha_l\kappa_l}\cdot\frac{\bar{M}_d^{\alpha_l}}{\Gamma^{\alpha_l-1}d^{2+\beta_2}b_w}\right]^{\frac{1}{\alpha_r+1}} \tag{68}$$

Step 4: GQA Ratio. From stationarity for gqa:

$$\mathrm{gqa}^{\alpha_m+1}=\frac{2\mu_T ld^{\alpha_m}}{\alpha_m\kappa_m}\left(d^2b_w+\bar{S}db_{kv}\right) \tag{69}$$

$$\mathrm{gqa}^*=\left[\frac{2\alpha_l\kappa_l}{\alpha_m\kappa_m}\cdot\frac{\Gamma^{\alpha_l-1}d^{\alpha_m}}{\bar{M}_d^{\alpha_l}}\left(d^2b_w+\bar{S}db_{kv}\right)\right]^{\frac{1}{\alpha_m+1}} \tag{70}$$

I.6 Solution Summary

$$\rho^*=\rho_{\min} \tag{71}$$

$$l^*=\frac{\bar{M}_d}{\xi_W^{\mathrm{dec}}d^2b_w+\frac{2\bar{S}db_{kv}}{\mathrm{gqa}}} \tag{72}$$

$$r^*=\left[\frac{\alpha_r\tilde{D}}{3\alpha_l\kappa_l}\cdot\frac{\bar{M}_d^{\alpha_l}}{\Gamma^{\alpha_l-1}d^{2+\beta_2}b_w}\right]^{\frac{1}{\alpha_r+1}} \tag{73}$$

$$\mathrm{gqa}^*=\left[\frac{2\alpha_l\kappa_l}{\alpha_m\kappa_m}\cdot\frac{\Gamma^{\alpha_l-1}d^{\alpha_m}}{\bar{M}_d^{\alpha_l}}\left(d^2b_w+\bar{S}db_{kv}\right)\right]^{\frac{1}{\alpha_m+1}} \tag{74}$$

where $\Gamma=\xi_W^{\mathrm{dec}}d^2b_w+2\bar{S}db_{kv}/\mathrm{gqa}$ and $\xi_W^{\mathrm{dec}}=2+2/\mathrm{gqa}^*+3r^*$ (implicit). The activation rate result corresponds to Theorem G.1 in the main text.

J Case D2: Decode, Memory-Constrained

J.1 Problem Statement

$$\begin{aligned} \min_{l,d,r,\mathrm{gqa},\rho} \quad & \hat{\mathcal{L}}(\theta) \\ \text{s.t.} \quad & l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w = M_{\text{budget}} \end{aligned} \tag{75}$$

where $\xi_W^{\text{all}} = 2 + 2/\mathrm{gqa} + 3r/\rho$.

Note: The memory constraint concerns model storage, which is independent of the KV-cache runtime overhead.

J.2 Lagrangian

$$\mathscr{L} = \hat{\mathcal{L}}(\theta) + \mu_M \left(l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w - M_{\text{budget}} \right) \tag{76}$$

J.3 Constraint Partial Derivatives

$$\frac{\partial g_M}{\partial l} = \xi_W^{\text{all}} d^2 b_w \tag{77}$$

$$\frac{\partial g_M}{\partial d} = 2l\xi_W^{\text{all}} d b_w \tag{78}$$

$$\frac{\partial g_M}{\partial r} = \frac{3ld^2 b_w}{\rho} \tag{79}$$

$$\frac{\partial g_M}{\partial \mathrm{gqa}} = -\frac{2ld^2 b_w}{\mathrm{gqa}^2} \tag{80}$$

$$\frac{\partial g_M}{\partial \rho} = -\frac{3lrd^2 b_w}{\rho^2} \tag{81}$$

J.4 KKT Conditions

Stationarity for l:

$$-\frac{\alpha_l \kappa_l}{l^{\alpha_l+1}} + \mu_M \xi_W^{\text{all}} d^2 b_w = 0 \quad \Rightarrow \quad \mu_M = \frac{\alpha_l \kappa_l}{l^{\alpha_l+1} \xi_W^{\text{all}} d^2 b_w} \tag{82}$$

Stationarity for ρ:

$$\frac{\alpha_\rho \kappa_\rho \rho^{\alpha_\rho - 1}}{r^{\alpha_r} d^{\beta_1}} - \frac{3\mu_M lrd^2 b_w}{\rho^2} = 0 \tag{83}$$

Rearranging:

$$\mu_M l = \frac{\alpha_\rho \kappa_\rho \rho^{\alpha_\rho + 1}}{3r^{\alpha_r+1} d^{\beta_1+2} b_w} \tag{84}$$

Stationarity for r:

$$-\frac{\alpha_r \tilde{D}}{r^{\alpha_r+1} d^{\beta_2}} + \frac{3\mu_M ld^2 b_w}{\rho} = 0 \tag{85}$$

Rearranging:

$$\mu_M l = \frac{\alpha_r \tilde{D} \rho}{3r^{\alpha_r+1} d^{\beta_2+2} b_w} \tag{86}$$

Stationarity for gqa:

$$\frac{\alpha_m \kappa_m \cdot \mathrm{gqa}^{\alpha_m - 1}}{d^{\alpha_m}} - \frac{2\mu_M ld^2 b_w}{\mathrm{gqa}^2} = 0 \tag{87}$$

J.5 Key Derivation: Activation Rate

Equation 84 and Equation 86:

$$\frac{\alpha_\rho \kappa_\rho \rho^{\alpha_\rho+1}}{3r^{\alpha_r+1}d^{\beta_1+2}b_w} = \frac{\alpha_r \tilde{D}\rho}{3r^{\alpha_r+1}d^{\beta_2+2}b_w} \tag{88}$$

Canceling common factors:

$$\frac{\alpha_\rho \kappa_\rho \rho^{\alpha_\rho}}{d^{\beta_1}} = \frac{\alpha_r \tilde{D}}{d^{\beta_2}} \tag{89}$$

Substituting $\tilde{D} = \kappa_\rho \rho^{\alpha_\rho} d^{\beta_2-\beta_1} + \kappa_d$:

$$\alpha_\rho \kappa_\rho \rho^{\alpha_\rho} d^{\beta_2-\beta_1} = \alpha_r \kappa_\rho \rho^{\alpha_\rho} d^{\beta_2-\beta_1} + \alpha_r \kappa_d \tag{90}$$

Collecting terms:

$$(\alpha_\rho - \alpha_r)\kappa_\rho \rho^{\alpha_\rho} d^{\beta_2-\beta_1} = \alpha_r \kappa_d \tag{91}$$

Solving:

$$\rho^* = \left[\frac{\alpha_r \kappa_d}{(\alpha_\rho - \alpha_r)\kappa_\rho}\right]^{1/\alpha_\rho} d^{(\beta_1-\beta_2)/\alpha_\rho} \tag{92}$$

Validity requires $\alpha_\rho > \alpha_r$.

J.6 Remaining Solutions

Depth.

$$l^* = \frac{M_{\text{budget}}}{\xi_W^{\text{all}} d^2 b_w} \tag{93}$$

Multiplier.

$$\mu_M l = \frac{\alpha_l \kappa_l (\xi_W^{\text{all}})^{\alpha_l-1} d^{2(\alpha_l-1)} b_w^{\alpha_l-1}}{M_{\text{budget}}^{\alpha_l}} \tag{94}$$

FFN Ratio.

$$r^* = \left[\frac{\alpha_r \tilde{D}\rho^*}{3\alpha_l \kappa_l} \cdot \frac{M_{\text{budget}}^{\alpha_l}}{(\xi_W^{\text{all}})^{\alpha_l-1} d^{2\alpha_l+\beta_2} b_w^{\alpha_l}}\right]^{\frac{1}{\alpha_r+1}} \tag{95}$$

GQA Ratio.

$$\text{gqa}^* = \left[\frac{2\alpha_l \kappa_l}{\alpha_m \kappa_m} \cdot \frac{(\xi_W^{\text{all}})^{\alpha_l-1} d^{2\alpha_l+\alpha_m} b_w^{\alpha_l}}{M_{\text{budget}}^{\alpha_l}}\right]^{\frac{1}{\alpha_m+1}} \tag{96}$$

J.7 Solution Summary

$$\rho^* = \left[\frac{\alpha_r \kappa_d}{(\alpha_\rho - \alpha_r)\kappa_\rho}\right]^{1/\alpha_\rho} d^{(\beta_1-\beta_2)/\alpha_\rho} \tag{97}$$

$$l^* = \frac{M_{\text{budget}}}{\xi_W^{\text{all}} d^2 b_w} \tag{98}$$

$$r^* = \left[\frac{\alpha_r \tilde{D}\rho^*}{3\alpha_l \kappa_l} \cdot \frac{M_{\text{budget}}^{\alpha_l}}{(\xi_W^{\text{all}})^{\alpha_l-1} d^{2\alpha_l+\beta_2} b_w^{\alpha_l}}\right]^{\frac{1}{\alpha_r+1}} \tag{99}$$

$$\text{gqa}^* = \left[\frac{2\alpha_l \kappa_l}{\alpha_m \kappa_m} \cdot \frac{(\xi_W^{\text{all}})^{\alpha_l-1} d^{2\alpha_l+\alpha_m} b_w^{\alpha_l}}{M_{\text{budget}}^{\alpha_l}}\right]^{\frac{1}{\alpha_m+1}} \tag{100}$$

where $\xi_W^{\text{all}} = 2 + 2/\text{gqa}^* + 3r^*/\rho^*$ (implicit). The activation rate result corresponds to Theorem G.2 in the main text.

K Case D3: Decode, Dual-Constrained

K.1 Problem Statement

$$\begin{aligned} \min_{l,d,r,\text{gqa},\rho} \quad & \hat{\mathcal{L}}(\theta) \\ \text{s.t.} \quad & l \cdot \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2l \cdot \bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} = \bar{M}_d \\ & l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w = M_{\text{budget}} \end{aligned} \tag{101}$$

K.2 Constraint Compatibility

Define:

$$\Gamma \triangleq \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2\bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} \tag{102}$$

From the two constraints:

$$l \cdot \Gamma = \bar{M}_d, \quad l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w = M_{\text{budget}} \tag{103}$$

Dividing:

$$\frac{\Gamma}{\xi_W^{\text{all}} \cdot d^2 \cdot b_w} = \frac{\bar{M}_d}{M_{\text{budget}}} \triangleq \eta \tag{104}$$

Expanding:

$$\frac{\xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2\bar{S} \cdot d \cdot b_{kv}}{\text{gqa}}}{\xi_W^{\text{all}} \cdot d^2 \cdot b_w} = \eta \tag{105}$$

$$\frac{\xi_W^{\text{dec}}}{\xi_W^{\text{all}}} + \frac{2\bar{S} \cdot b_{kv}}{\xi_W^{\text{all}} \cdot \text{gqa} \cdot d \cdot b_w} = \eta \tag{106}$$

K.3 Derivation of Activation Rate

Substituting $\xi_W^{\text{dec}} = \alpha_{\text{attn}} + 3r$ and $\xi_W^{\text{all}} = \alpha_{\text{attn}} + 3r/\rho$ where $\alpha_{\text{attn}} = 2 + 2/\text{gqa}$:

$$\frac{\alpha_{\text{attn}} + 3r}{\alpha_{\text{attn}} + \frac{3r}{\rho}} + \frac{2\bar{S} \cdot b_{kv}}{\left(\alpha_{\text{attn}} + \frac{3r}{\rho}\right) \text{gqa} \cdot d \cdot b_w} = \eta \tag{107}$$

Define the KV-cache correction term:

$$\delta \triangleq \frac{2\bar{S} \cdot b_{kv}}{\text{gqa} \cdot d \cdot b_w} \tag{108}$$

Then:

$$\frac{\alpha_{\text{attn}} + 3r + \delta}{\xi_W^{\text{all}}} = \eta. \tag{109}$$

Hence:

$$\xi_W^{\text{all}} = \frac{\alpha_{\text{attn}} + 3r + \delta}{\eta}. \tag{110}$$

Substituting $\xi_W^{\text{all}} = \alpha_{\text{attn}} + 3r/\rho$ gives

$$\rho^* = \frac{3\eta r}{\alpha_{\text{attn}}(1 - \eta) + 3r + \delta}. \tag{111}$$

For fixed remaining variables, this is a compatibility relation, not a quadratic equation. For $r > 0$ and $\eta > 0$, enforce $\rho_{\min} \le \rho^* \le 1$; the upper bound requires $\eta \le 1 + \delta/(\alpha_{\text{attn}} + 3r)$. The remaining KKT conditions and discrete feasibility must also hold.

K.4 Special Case: $\delta \to 0$

When the KV-cache term is negligible ($\delta \to 0$):

$$\xi_W^{\text{all}} \to \frac{\alpha_{\text{attn}} + 3r}{\eta} \tag{112}$$

$$\rho^* \to \frac{3\eta r}{\alpha_{\text{attn}} + 3r - \eta\alpha_{\text{attn}}} = \frac{3\eta r}{\alpha_{\text{attn}}(1-\eta) + 3r} \tag{113}$$

This recovers the simplified formula from the weight-dominated approximation.

K.5 Remaining Solutions

Depth.

$$l^* = \frac{M_{\text{budget}}}{\xi_W^{\text{all}} d^2 b_w} \tag{114}$$

Other Variables. Unlike the single-constraint cases (D1, D2, P1), where a single Lagrange multiplier allows sequential elimination, the dual-constrained case involves two active constraints with multipliers μ_T and μ_M. The stationarity conditions for r and gqa each depend on both multipliers, and the KV-cache term in the decode constraint further couples gqa to the latency budget. As a result, r^*, gqa^*, and d^* do not admit independent closed-form expressions and must be obtained by numerically solving the coupled KKT system. In practice, ρ^* and l^* from the equations above are substituted first, reducing the system to three unknowns.

K.6 Solution Summary

$$\eta = \bar{M}_d / M_{\text{budget}} \tag{115}$$

$$\delta = \frac{2\bar{S} b_{kv}}{\text{gqa} \cdot d \cdot b_w} \tag{116}$$

$$\xi_W^{\text{all}} = \frac{\alpha_{\text{attn}} + 3r + \delta}{\eta} \tag{117}$$

$$\rho^* = \frac{3r}{\xi_W^{\text{all}} - \alpha_{\text{attn}}} \tag{118}$$

$$l^* = \frac{M_{\text{budget}}}{\xi_W^{\text{all}} d^2 b_w} \tag{119}$$

where $\alpha_{\text{attn}} = 2 + 2/\text{gqa}^*$, and r^*, gqa^* are coupled solutions. The activation rate result corresponds to Theorem G.4(b) in the main text.

L Case P1: Prefill, Latency-Constrained

L.1 Problem Statement

$$\begin{aligned} \min_{l,d,r,\text{gqa},\rho} \quad & \hat{\mathcal{L}}(\theta) \\ \text{s.t.} \quad & l \cdot \xi_F \cdot d^2 = \bar{F}_p \\ & \rho \geq \rho_{\min} \end{aligned} \tag{120}$$

where $\xi_F = 4 + 4/\text{gqa} + 6r$.

Note: The prefill constraint is compute-bound and does not involve KV-cache traffic.

L.2 Lagrangian

$$\mathscr{L} = \hat{\mathcal{L}}(\theta) + \mu_T \left(l \cdot \xi_F \cdot d^2 - \bar{F}_p \right) \tag{121}$$

L.3 Constraint Partial Derivatives

$$\frac{\partial g_T}{\partial l} = \xi_F d^2 \tag{122}$$

$$\frac{\partial g_T}{\partial d} = 2l\xi_F d \tag{123}$$

$$\frac{\partial g_T}{\partial r} = 6ld^2 \tag{124}$$

$$\frac{\partial g_T}{\partial \mathrm{gqa}} = -\frac{4ld^2}{\mathrm{gqa}^2} \tag{125}$$

$$\frac{\partial g_T}{\partial \rho} = 0 \tag{126}$$

L.4 KKT Conditions

Stationarity for l:

$$-\frac{\alpha_l \kappa_l}{l^{\alpha_l+1}} + \mu_T \xi_F d^2 = 0 \quad \Rightarrow \quad \mu_T = \frac{\alpha_l \kappa_l}{l^{\alpha_l+1} \xi_F d^2} \tag{127}$$

Stationarity for ρ:

$$\frac{\partial \mathscr{L}}{\partial \rho} = \frac{\alpha_\rho \kappa_\rho \rho^{\alpha_\rho - 1}}{r^{\alpha_r} d^{\beta_1}} > 0 \tag{128}$$

Since $\partial g_T / \partial \rho = 0$ and $\partial \hat{\mathcal{L}} / \partial \rho > 0$:

$$\rho^* = \rho_{\min} \tag{129}$$

Stationarity for r:

$$-\frac{\alpha_r \tilde{D}}{r^{\alpha_r+1} d^{\beta_2}} + 6\mu_T l d^2 = 0 \tag{130}$$

Note: coefficient is 6 (vs 3 in D1) from $\partial \xi_F / \partial r = 6$. The b_w factor is absent because the prefill constraint operates in FLOPs rather than bytes, yielding $\partial g_T / \partial r = 6ld^2$ without the byte-width scaling present in the decode case ($\partial g_T / \partial r = 3ld^2 b_w$).

Stationarity for gqa:

$$\frac{\alpha_m \kappa_m \cdot \mathrm{gqa}^{\alpha_m - 1}}{d^{\alpha_m}} - \frac{4\mu_T l d^2}{\mathrm{gqa}^2} = 0 \tag{131}$$

Note: coefficient is 4 (vs 2 in D1) from $|\partial \xi_F / \partial \mathrm{gqa}| = 4/\mathrm{gqa}^2$.

L.5 Solution Derivation

Depth.

$$l^* = \frac{\bar{F}_p}{\xi_F d^2} \tag{132}$$

Multiplier.

$$\mu_T l = \frac{\alpha_l \kappa_l \xi_F^{\alpha_l - 1} d^{2(\alpha_l - 1)}}{\bar{F}_p^{\alpha_l}} \tag{133}$$

FFN Ratio.

$$r^* = \left[\frac{\alpha_r \tilde{D}}{6\alpha_l \kappa_l} \cdot \frac{\bar{F}_p^{\alpha_l}}{\xi_F^{\alpha_l - 1} d^{2\alpha_l + \beta_2}} \right]^{\frac{1}{\alpha_r + 1}} \tag{134}$$

GQA Ratio.

$$\mathrm{gqa}^* = \left[\frac{4\alpha_l \kappa_l}{\alpha_m \kappa_m} \cdot \frac{\xi_F^{\alpha_l - 1} d^{2\alpha_l + \alpha_m}}{\bar{F}_p^{\alpha_l}} \right]^{\frac{1}{\alpha_m + 1}} \tag{135}$$

L.6 Comparison with Case D1

Table 18: Comparison of prefill (P1) and decode (D1) phase optimal hyperparameters. The listed prefactors do not by themselves determine the ordering of optimal expansion or GQA ratios.

	P1 (Prefill)	**D1 (Decode)**
r^* coefficient	$1/6$	$1/3$
gqa^* coefficient	4	2
Constraint	ξ_F	ξ_W^{dec} + KV term

Table 18 compares derivative-related prefactors. The full expressions also depend on budgets, coupled architecture variables, and the decode KV-cache term. Therefore, the factors $1/6$ versus $1/3$ and 4 versus 2 do not alone imply a larger or smaller optimum.

L.7 Solution Summary

$$\rho^* = \rho_{\min} \tag{136}$$

$$l^* = \frac{\bar{F}_p}{\xi_F d^2} \tag{137}$$

$$r^* = \left[\frac{\alpha_r \tilde{D}}{6\alpha_l \kappa_l} \cdot \frac{\bar{F}_p^{\alpha_l}}{\xi_F^{\alpha_l - 1} d^{2\alpha_l + \beta_2}} \right]^{\frac{1}{\alpha_r + 1}} \tag{138}$$

$$\mathrm{gqa}^* = \left[\frac{4\alpha_l \kappa_l}{\alpha_m \kappa_m} \cdot \frac{\xi_F^{\alpha_l - 1} d^{2\alpha_l + \alpha_m}}{\bar{F}_p^{\alpha_l}} \right]^{\frac{1}{\alpha_m + 1}} \tag{139}$$

where $\xi_F = 4 + 4/\mathrm{gqa}^* + 6r^*$ (implicit). The activation rate result corresponds to Theorem G.1 in the main text.

M Case P2: Prefill, Memory-Constrained

M.1 Problem Statement

$$\begin{aligned} \min_{l,d,r,\mathrm{gqa},\rho} \quad & \hat{\mathcal{L}}(\theta) \\ \text{s.t.} \quad & l \cdot \xi_W^{\mathrm{all}} \cdot d^2 \cdot b_w = M_{\mathrm{budget}} \end{aligned} \tag{140}$$

where $\xi_W^{\mathrm{all}} = 2 + 2/\mathrm{gqa} + 3r/\rho$.

M.2 Equivalence to Case D2

The Lagrangian is:

$$\mathscr{L} = \hat{\mathcal{L}}(\theta) + \mu_M \left(l \cdot \xi_W^{\mathrm{all}} \cdot d^2 \cdot b_w - M_{\mathrm{budget}} \right) \tag{141}$$

This is identical to Case D2 because:

1. The loss function $\hat{\mathcal{L}}(\theta)$ is independent of scenario
2. The memory constraint concerns model storage, not runtime behavior
3. ρ appears only in ξ_W^{all}

All KKT conditions and solutions are identical to Case D2.

M.3 Solution Summary

$$\rho^* = \left[\frac{\alpha_r \kappa_d}{(\alpha_\rho - \alpha_r)\kappa_\rho}\right]^{1/\alpha_\rho} d^{(\beta_1-\beta_2)/\alpha_\rho} \tag{142}$$

$$l^* = \frac{M_{\text{budget}}}{\xi_W^{\text{all}} d^2 b_w} \tag{143}$$

$$r^* = \left[\frac{\alpha_r \tilde{D}\rho^*}{3\alpha_l \kappa_l} \cdot \frac{M_{\text{budget}}^{\alpha_l}}{(\xi_W^{\text{all}})^{\alpha_l - 1} d^{2\alpha_l+\beta_2} b_w^{\alpha_l}}\right]^{\frac{1}{\alpha_r+1}} \tag{144}$$

$$\text{gqa}^* = \left[\frac{2\alpha_l \kappa_l}{\alpha_m \kappa_m} \cdot \frac{(\xi_W^{\text{all}})^{\alpha_l - 1} d^{2\alpha_l+\alpha_m} b_w^{\alpha_l}}{M_{\text{budget}}^{\alpha_l}}\right]^{\frac{1}{\alpha_m+1}} \tag{145}$$

All solutions identical to Case D2. The activation rate result corresponds to Theorem G.2 in the main text.

N Case P3: Prefill, Dual-Constrained

N.1 Problem Statement

$$\begin{aligned} \min_{l,d,r,\text{gqa},\rho} \quad & \hat{\mathcal{L}}(\theta) \\ \text{s.t.} \quad & l \cdot \xi_F \cdot d^2 = \bar{F}_p \\ & l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w = M_{\text{budget}} \end{aligned} \tag{146}$$

Note: The prefill constraint is compute-bound and does not involve KV-cache.

N.2 Constraint Compatibility

Dividing the two constraints:

$$\frac{\xi_F}{b_w \cdot \xi_W^{\text{all}}} = \frac{\bar{F}_p}{M_{\text{budget}}} \triangleq \eta_p \tag{147}$$

N.3 Derivation of Activation Rate

Using $\xi_F = 2(\alpha_{\text{attn}} + 3r)$ and $\xi_W^{\text{all}} = \alpha_{\text{attn}} + 3r/\rho$:

$$\frac{2(\alpha_{\text{attn}} + 3r)}{b_w(\alpha_{\text{attn}} + \frac{3r}{\rho})} = \eta_p \tag{148}$$

Cross-multiplying:

$$2\alpha_{\text{attn}} + 6r = \eta_p b_w \alpha_{\text{attn}} + \frac{3\eta_p b_w r}{\rho} \tag{149}$$

Rearranging:

$$\alpha_{\text{attn}}(2 - \eta_p b_w) + 6r = \frac{3\eta_p b_w r}{\rho} \tag{150}$$

Solving:

$$\rho^* = \frac{3\eta_p b_w r}{\alpha_{\text{attn}}(2 - \eta_p b_w) + 6r} \tag{151}$$

where $\eta_p = \bar{F}_p / M_{\text{budget}}$ and $\alpha_{\text{attn}} = 2 + 2/\text{gqa}$.

For $r > 0$, feasibility with $0 < \rho \leq 1$ requires $0 < \eta_p b_w \leq 2$; equality gives $\rho = 1$. The lower grid bound $\rho \geq \rho_{\min}$ must also hold. This result corresponds to Theorem G.4(a) in the main text.

N.4 Comparison with Case D3

Table 19: Comparison of prefill (P3) and decode (D3) phase characteristics. Decode includes KV-cache memory access costs; both columns give conditional relations for the activation rate ρ^*.

	P3 (Prefill)	**D3 (Decode)**
η definition	$\bar{F}_p / M_{\text{budget}}$	$\bar{M}_d / M_{\text{budget}}$
Latency constraint	$l\xi_F d^2$	$l(\xi_W^{\text{dec}} d^2 b_w + \frac{2\bar{S}db_{kv}}{\text{gqa}})$
KV-cache term	absent	present
ρ^* formula	closed-form	conditional closed-form

As shown in Table 19, the key difference between prefill and decode phases lies in the memory access patterns and their impact on the optimal architecture. In prefill (P3), the latency constraint depends only on compute ($\xi_F d^2$) and admits a closed-form solution for ρ^*. In decode (D3), the latency constraint includes both weight loading and KV-cache access, with the KV-cache term $\frac{2Sdb_{kv}}{\text{gqa}}$ becoming dominant for large sequence lengths. The KV-cache correction changes the conditional activation-rate expression; the remaining variables still require the coupled KKT conditions.

N.5 Remaining Solutions

Depth.

$$l^* = \frac{\bar{F}_p}{\xi_F d^2} \tag{152}$$

Other Variables. The solutions for r^*, gqa^*, d^* require solving the coupled KKT system.

N.6 Solution Summary

$$\eta_p = \bar{F}_p / M_{\text{budget}}, \quad 0 < \eta_p b_w \leq 2 \tag{153}$$

$$\rho^* = \frac{3\eta_p b_w r^*}{\alpha_{\text{attn}}(2 - \eta_p b_w) + 6r^*} \tag{154}$$

$$l^* = \frac{\bar{F}_p}{\xi_F d^2} \tag{155}$$

where $\alpha_{\text{attn}} = 2 + 2/\text{gqa}^*$, and r^*, gqa^* are coupled solutions. The activation rate result corresponds to Theorem G.4(a) in the main text.

O Summary

O.1 Constraint Forms

The constraint forms are shown in Table 20.

Table 20: Constraint formulations.

Constraint	Formula
Prefill latency	$l \cdot \xi_F \cdot d^2 \leq \bar{F}_p$
Decode latency	$l \cdot \xi_W^{\text{dec}} \cdot d^2 \cdot b_w + \frac{2l \cdot \bar{S} \cdot d \cdot b_{kv}}{\text{gqa}} \leq \bar{M}_d$
Memory	$l \cdot \xi_W^{\text{all}} \cdot d^2 \cdot b_w \leq M_{\text{budget}}$

Table 21: Solution Matrix: Activation Rate ρ^*.

	Latency	Memory	Dual
Decode	$\rho_{\min}$	$\left[\frac{\alpha_r \kappa_d}{(\alpha_\rho - \alpha_r)\kappa_\rho}\right]^{1/\alpha_\rho} d^{\frac{\beta_1 - \beta_2}{\alpha_\rho}}$	$\frac{3\eta r}{\alpha_{\text{attn}}(1-\eta)+3r+\delta}$
Prefill	$\rho_{\min}$	same as Decode	$\frac{3\eta_p b_w r}{\alpha_{\text{attn}}(2-\eta_p b_w)+6r}$

O.2 Solution Matrix: Activation Rate ρ^*

The activation rate ρ^* in solution matrix are shown in Table 21.

O.3 Solution Matrix: Depth l^*

The Depth l^* in solution matrix are shown in Table 22.

Table 22: Solution Matrix: Depth l^*.

	Latency / Dual	Memory
Decode	$\frac{\bar{M}_d}{\xi_W^{\text{dec}} d^2 b_w + \frac{2\bar{S} d b_{kv}}{\text{gqa}}}$	$\frac{M_{\text{budget}}}{\xi_W^{\text{all}} d^2 b_w}$
Prefill	$\frac{\bar{F}_p}{\xi_F d^2}$	$\frac{M_{\text{budget}}}{\xi_W^{\text{all}} d^2 b_w}$

O.4 Coefficient Comparison

Table 23 lists extracted prefactors, not complete solutions. Budgets and coupled terms differ across regimes; these factors alone do not order the optimal ratios.

Table 23: Coefficient Comparison.

	r^* coefficient		gqa* coefficient	
	Prefill	Decode	Prefill	Decode
Latency	$1/6$	$1/3$	4	2
Memory	$\rho/3$	$\rho/3$	2	2

O.5 Key Results

1. Width-Sparsity Scaling. Under memory constraint:

$$\rho^* = \left[\frac{\alpha_r \kappa_d}{(\alpha_\rho - \alpha_r)\kappa_\rho}\right]^{1/\alpha_\rho} d^{(\beta_1 - \beta_2)/\alpha_\rho} \tag{156}$$

2. Scenario Independence. Memory-constrained ρ^* is identical for Prefill and Decode.

3. Prefill-Decode Asymmetry. Latency-constrained coefficients differ since $\partial \xi_F / \partial r = 6$, comparing to $\partial \xi_W^{\text{dec}} / \partial r = 3$.

4. KV-Cache Effect. The decode latency constraint includes the KV-cache term $2l\bar{S}db_{kv}/\mathrm{gqa}$, which:

- Increases effective constraint tightness
- Couples gqa more strongly to the constraint
- Enters additively in the D3 compatibility numerator

O.6 Notation

Table 24: Notation Extension for Appendices.

ξ_F	$4 + 4/\mathrm{gqa} + 6r$	ξ_W^{dec}	$2 + 2/\mathrm{gqa} + 3r$
ξ_W^{all}	$2 + 2/\mathrm{gqa} + 3r/\rho$	$\bar{D}$	$\kappa_\rho \rho^{\alpha_\rho} d^{\beta_2 - \beta_1} + \kappa_d$
α_{attn}	$2 + 2/\mathrm{gqa}$	$\bar{S}$	average context length
$\bar{F}_p$	$T_{\mathrm{lat}}\pi_H/(BS_{\mathrm{in}})$	$\bar{M}_d$	$T_{\mathrm{lat}}\beta_H/S_{\mathrm{out}}$
η	$\bar{M}_d/M_{\mathrm{budget}}$	η_p	$\bar{F}_p/M_{\mathrm{budget}}$
Γ	$\xi_W^{\mathrm{dec}} d^2 b_w + 2\bar{S}db_{kv}/\mathrm{gqa}$		

Here we present the notation Table 24 used in the Appendices, as an extension to Table 7.

P Deployment Workflow for New Hardware

For a new platform with specifications $(\pi_H, \beta_H, M_{\text{budget}})$:

1. **Measure hardware parameters:** peak compute π_H, sustained memory bandwidth β_H, available memory M_{budget}.
2. **Define application requirements:** target latency budgets $T_{\text{lat}}^{\text{pre}}, T_{\text{lat}}^{\text{dec}}$ and workload $(B, S_{\text{in}}, S_{\text{out}})$.
3. **Compute normalized budgets:** $\bar{F}_p = T_{\text{lat}}^{\text{pre}} \cdot \pi_H/(BS_{\text{in}})$ and $\bar{M}_d = T_{\text{lat}}^{\text{dec}} \cdot \beta_H/S_{\text{out}}$.
4. **Determine regime:** use the budget ratios as compatibility parameters, then evaluate candidate feasibility and which constraints are active. A ratio alone does not determine the active regime; the prefill ratio must be considered with b_w to form a dimensionless quantity.
5. **Apply theorem:** obtain conditional continuous candidates, then evaluate feasible discrete configurations and boundary cases.
6. **Validate:** profile latency under the target workload; assess quality through separate training and evaluation.

Example. For bandwidth 50 GB/s, memory budget 4 GB, and a total decode budget of 100 ms over $S_{\mathrm{out}} = 16$ modeled passes, $\bar{M}_d = 0.3125$ GB and $\eta = 0.078125$. These values do not by themselves establish the active constraint or a model depth. If the interior memory-constrained conditions apply, $d = 1024$ gives $\rho^* \approx 0.4024$ under Appendix C; feasible expert counts and depth still require evaluation.

Q Limitations and Future Extensions

The empirical loss function is fitted under a fixed 10B-token training recipe. Its accuracy is established by the reported validation, rather than assumed for arbitrary architectures, datasets, or token budgets. A new deployment setting may require additional calibration data. Linear-attention and state-space architectures are not covered by the current parameterization.

Roofline modeling does not fully capture kernel-launch overhead, cache behavior, or operator fusion. Prediction errors therefore depend on the implementation and workload; target-device measurements remain necessary for validating selected candidates. Searches on additional hardware illustrate model predictions rather than measured cross-platform performance.

The reported frontiers are conditional on the search grid and the fitted objectives. Subsequent training and profiling can refine the ranking of retained candidates, but cannot recover promising architectures discarded during screening without expanding or repeating the search. Precision-specific latency and memory estimates do not include quantization-induced quality changes, which require separate evaluation.

R Multi-Device Architecture Selection

Fix the training recipe, precision, and inference workload. Let Θ_H be the feasible architecture set on device H, and define its predicted loss envelope at latency budget t as

$$L_H^*(t) = \min_{\theta' \in \Theta_H : \widehat{T}_H(\theta') \le t} \widehat{L}(\theta').$$

For tolerance $\delta \ge 0$, define the relaxed Pareto band

$$B_H(\delta) = \{\theta \in \Theta_H : \widehat{L}(\theta) - L_H^*(\widehat{T}_H(\theta)) \le \delta\}.$$

A candidate in $\bigcap_H B_H(\delta)$ is feasible on every target device and is within δ of each device's predicted loss envelope at its own latency. This is a per-device comparison, not a claim of global optimality across devices. If the intersection is empty, a larger tolerance may admit candidates when a common feasible architecture exists. Otherwise the deployment requirements or device-specific architecture choices must be reconsidered. This construction is a proposed extension.

S The Use of Large Language Models

LLMs assisted with language editing, manuscript consistency checks, code and formula review, and suggestions for revising the presentation. These tools do not constitute experimental evidence. The authors retain responsibility for the scientific claims, interpretation of results, and final manuscript.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state four contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix Q

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The assumptions and proofs of the theorems are provided in Appendix G and Appendix I-N.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The search space and training settings are described in Appendix A and Appendix B. A complete configuration and artifact manifest for reproducing all reported experiments is not provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: This manuscript does not provide a complete public package of the code, training data, measurement records, and checkpoints needed to reproduce all results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [No]

Justification: The paper reports the main training and evaluation settings in Section 4.1 and Appendix B, but does not provide a complete per-run configuration and data manifest.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The frontier and matched-latency perplexity results are point estimates. Training-seed confidence intervals and statistical significance tests are not reported; variability across repeated latency measurements is distinct from uncertainty across training runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Appendix B reports a calibration-training estimate, but a complete per-stage resource accounting for all experiments is not provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research has been conducted in accordance with the NeurIPS Code of Ethics. The work involves no human subjects, no private user data, no scraped personal information, and no high-risk model release.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: On-device deployment (Section 1) brings positive impacts in energy efficiency, on-device privacy, and accessibility; the methodology introduces no risks beyond those of existing on-device LLM systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper proposes an analytical framework, not a high-capability model release; the trained sub-billion-parameter architecture poses no elevated misuse risk.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: The paper cites the models, datasets, and tools used or discussed, but does not provide a complete version-specific license inventory. Public availability alone does not establish permission for every use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This manuscript does not link a new public asset release.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work involves no human subjects; IRB review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the research subject and also assisted manuscript preparation, as described in Appendix S.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.