

Complexity-Aware Routing: Detecting and Allocating Dynamic Compute for Efficient Mixture-of-Experts Large Language Models

Jiahe Li¹, Saisai Hu², Menghao Li¹, Shaolin Zhu^{1*}, Deyi Xiong¹

¹TJUNLP Lab, School of Computer Science and Technology, Tianjin University, China

²Department of Computer Science, Pace University, United States

{3023244389, zhushaolin, dyxiong}@tju.edu.cn

Abstract

Current Mixture-of-Experts (MoE) models typically rely on a static Top- k routing mechanism, activating a fixed number of experts for every token regardless of its difficulty. This uniform allocation ignores the inherent variability in linguistic information density that simple functional words require significantly less processing capacity than content-heavy terms. The static routing results in computational waste for easy tokens and potential under-fitting for difficult ones. To address this issue, we propose **CAR**, a Complexity-Aware Routing framework that aligns the computational budget with the intrinsic information density of input tokens. Diverging from previous dynamic methods that rely on unstable heuristics, we ground our routing decision in a dual-signal objective combining Shannon Surprisal and Predictive Entropy to objectively measure token complexity. We implement a decoupled architecture where a Complexity Perception (CP) module utilizes a Recurrent Complexity Memory (RCM) to estimate these signals from a teacher model, guiding a Dynamic Expert Allocator (DEA) to adjust k on the fly. Extensive experiments demonstrate that CAR consistently outperforms both static baselines and SOTA dynamic routing methods. Notably, our approach achieves superior accuracy on reasoning-intensive benchmarks (e.g., boosting MMLU by 0.8 and GSM8K by 1.5 over static routing) while reducing the average number of activated parameters by approximately 10%. Our code is available at <https://github.com/Acking646/car>.

1 Introduction

The scaling of Large Language Models (LLMs) has driven remarkable capabilities across a spectrum of natural language processing tasks (Touvron et al., 2023; Zhu et al., 2024; Shen et al., 2025; Dong et al., 2025). However, the immense

computational cost associated with dense models has catalyzed the adoption of Sparse Mixture-of-Experts (MoE) architectures (Shazeer et al., 2017; Fedus et al., 2022). By conditionally activating only a subset of parameters, also known as experts, for each input token, MoEs promise to decouple model capacity from computational cost, achieving performance comparable to dense models with significantly fewer activated parameters (Jiang et al., 2024; Dai et al., 2024).

Despite these advances, the standard routing mechanism in MoE models typically employs a static Top- k strategy, where a fixed number of experts (e.g., $k=2$) is assigned to every token regardless of its linguistic complexity. This one-size-fits-all approach ignores the inherent variability in information density across tokens: a functional stop-word like “the” requires significantly less computational capacity than a content-heavy word like “ontology” or a complex reasoning step. Consequently, static routing leads to a dual problem: computational waste for simple tokens and potential under-fitting for complex tokens that might benefit from broader expert collaboration (Aghdam et al., 2024; Raposo et al., 2024).

To address the rigidity of fixed computation, recent studies have explored dynamic routing mechanisms. Approaches such as MoD (Raposo et al., 2024) and DynMoE (Guo et al., 2024) attempt to allocate compute dynamically by allowing variable expert activation or skipping layers entirely. Similarly, LD-MoLE (Zhuang et al., 2026) and MoNE (Cheng et al., 2025) investigate finer-grained or learnable sparsity to optimize parameter utilization. Furthermore, recent paradigms like Ada-K Routing (Zhao et al., 2025) frame dynamic expert selection as a reinforcement learning problem, optimizing the routing policy directly based on downstream task rewards. However, these methods often rely on heuristic thresholds (e.g., Top- p routing), end-to-end learnable gates,

*Corresponding author.

or reward-driven optimizations that suffer from training instability and lack interpretability. Crucially, existing dynamic routers typically derive routing decisions implicitly from the model’s internal confidence, task-specific downstream objectives, or load-balancing constraints, which do not explicitly measure the intrinsic complexity of the input token.

To address these issues, we propose Complexity-Aware Routing (CAR), a novel framework that fundamentally rethinks expert allocation by introducing objective, information-theoretic signals into the routing decision. Diverging from heuristic-based methods, we ground our approach in a dual-signal framework comprising Shannon Surprisal (Oh and Schuler, 2025) and Predictive Entropy (Malinin and Gales, 2020). This posits that the computational budget allocated to a token should be proportional to both its contextual unpredictability and the overall uncertainty of the next-token distribution.

We implement CAR through a Perception-Decision decoupling strategy. We first construct a Complexity Perception (CP) module utilizing cross-layer recurrence, trained to regress dual complexity signals generated by a high-capacity teacher model. This perception allows the student MoE model to explicitly perceive the complexity of incoming tokens. Then, we design a discrete Dynamic Expert Allocator (DEA) decision that maps these signals to a dynamic k that utilizes a robust normalization strategy to handle the long-tail distribution of complexity values. This design enables the model to sparsely activate as few as one expert for simple tokens while engaging a broader ensemble for complex ones, optimizing the trade-off between performance and efficiency.

Extensive experiments demonstrate that CAR significantly outperforms both standard Top- k baselines and existing dynamic routing methods. Specifically, CAR achieves superior performance on reasoning-intensive benchmarks (e.g., boosting MMLU accuracy by 0.8 and GSM8K by 1.5 over static routing) while reducing the average number of activated parameters by approximately 10%, validating that explicitly modeling token complexity is a more effective path toward efficient, adaptive computing in LLMs.

The main contributions of this work are summarized as follows:

- We identify the limitation of static compute

allocation in MoEs and propose using Shannon Surprisal and Predictive Entropy to guide dynamic expert routing.

- We introduce the CAR framework, which decouples complexity perception from routing decisions. It uses shared recurrence to propagate token states and a regression-based predictor to estimate dual complexity signals, stabilizing the dynamic routing process without introducing training collapse.
- We demonstrate through extensive experiments that CAR achieves a better accuracy-efficiency trade-off compared with existing baselines.

2 Related Work

Mixture of Experts The concept of MoEs has been initially proposed to decompose complex tasks into sub-tasks handled by specialized sub-networks (Jacobs et al., 1991). Shazeer et al. (2017) introduce the Sparsely-Gated MoE layer that employs a trainable gating network to route tokens to a sparse subset of experts to scale model capacity without a proportional increase in computational cost. This architecture has become a cornerstone for scaling large language models. Representative works such as GShard (Lepikhin et al., 2020) and Switch Transformer (Fedus et al., 2022) demonstrated the efficacy of MoE in scaling to trillions of parameters by selecting the top-1 or top-2 experts per token. More recently, architectures like Mixtral (Jiang et al., 2024) and DeepSeek-MoE (Dai et al., 2024) have further refined expert specialization and granularity, establishing MoE as a standard for high-performance, efficient LLMs. MoE-style architectures are also prevalent in multilingual LLMs and machine translation (Li et al., 2023; Zhu et al., 2025; Li et al., 2026; Dong et al., 2026). However, these conventional approaches typically rely on a static Top- k routing mechanism, which rigidly assigns a fixed computational budget to every token regardless of its information density or processing difficulty.

Dynamic Routing Mechanisms To address the inefficiencies of static routing, recent research has focused on dynamic expert allocation strategies that adapt to token complexity (Zhou et al., 2022; Huang et al., 2024; Raposo et al., 2024; Guo et al., 2024; Zhuang et al., 2026; Cheng et al., 2025). Early attempts like ECR (Zhou et al., 2022) invert

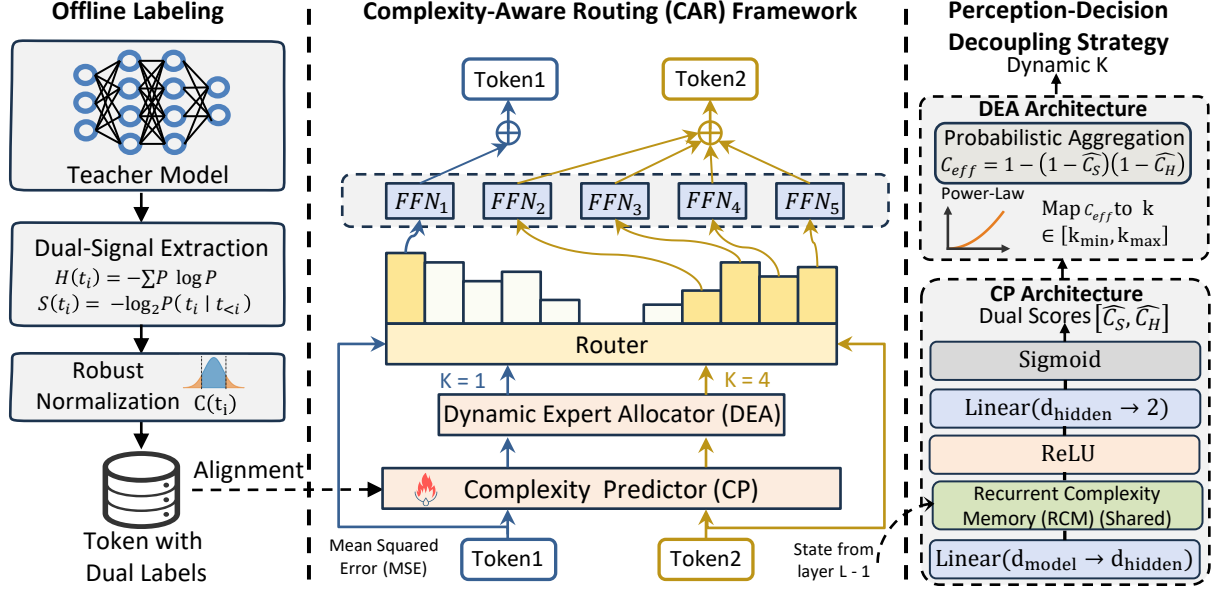


Figure 1: Overview of the Complexity-Aware Routing (CAR) framework. (i) **Offline labeling**: a frozen teacher model produces token-level dual complexity labels from Shannon Surprisal (S) and Predictive Entropy (H), followed by robust normalization. (ii) **Predictor alignment**: the normalized labels are used to train the Complexity Perception (CP) module with Recurrent Complexity Memory (RCM), while the MoE backbone remains frozen. (iii) **Dynamic inference**: CP predicts token-level complexity scores, and the Dynamic Expert Allocator (DEA) maps the effective complexity score C_{eff} to a dynamic expert budget $k \in [k_{\min}, k_{\max}]$.

the selection process to prioritize expert load balancing, while Top- p routing (Huang et al., 2024) introduces probability thresholds to allow variable expert activation. MoD (Raposo et al., 2024) and DynMoE (Guo et al., 2024) dynamically allocate compute by skipping layers or adjusting expert counts during training. Similarly, AdaMoE (Zeng et al., 2024) and MoE++ (Jin et al., 2025) attempt to identify easy tokens to reduce computation via null experts. LD-MoLE (Zhuang et al., 2026) and MoNE (Cheng et al., 2025) further explore learnable routing distributions and neuron-level sparsity. Additionally, Ada-K Routing (Zhao et al., 2025) formulates expert allocation as a reinforcement learning problem, optimizing routing policies from downstream task rewards. While this design can directly optimize task-specific objectives, the learned policy may depend on the reward design and task distribution used during policy training. More broadly, existing dynamic routing methods often derive allocation decisions from heuristic confidence scores, router probabilities, or downstream reward feedback, which do not explicitly measure token-level complexity. In contrast, CAR uses task-agnostic, teacher-derived complexity signals to supervise a lightweight predictor under a frozen-backbone setting. By regressing Shannon Surprisal and Predictive Entropy, CAR

separates complexity estimation from the original router confidence and downstream reward optimization, enabling a more interpretable basis for dynamic expert allocation.

3 Methodology

The proposed CAR framework is illustrated in Figure 1. We first establish the theoretical basis for utilizing a dual-signal objective—combining Shannon Surprisal and Predictive Entropy—to capture token complexity (Section 3.1). Then, we detail the automated labeling pipeline, which employs a frozen teacher model to generate complexity scores via efficient forward propagation (Section 3.2). Finally, we present the CAR architecture, which utilizes a shared cross-layer recurrent mechanism coupled with a Complexity Perception (CP) predictor and a Dynamic Expert Allocator (DEA) to decouple continuous complexity estimation from discrete expert allocation (Section 3.3). We then describe the predictor alignment phase, which ensures stable routing initialization under a frozen-backbone setting (Section 3.4).

3.1 Quantifying Token Complexity via Dual Signals

A foundational requirement for the proposed CAR is a reliable and quantifiable metric that accu-

rately reflects token-level complexity. To address this, we draw from information theory and employ a dual-signal framework comprising Shannon Surprisal (Oh and Schuler, 2025) and Predictive Entropy (Malinin and Gales, 2020). Grounded in cognitive linguistics (Gruteke Klein et al., 2025; Oh and Schuler, 2025), surprisal correlates strongly with the processing effort required for language processing. Furthermore, recent studies on cognitive-load-aware activation (Yang et al., 2026) demonstrate that surprisal and predictive entropy independently and complementarily predict cognitive processing demand. Crucially, unlike heuristic constraints, these dual signals provide objective, information-theoretic measures of processing difficulty.

Formally, let $\mathbf{M}_{\text{teacher}}$ denote a robust, frozen pre-trained teacher language model parameterized by Θ_T . For a text sequence $\mathbf{x} = (t_1, t_2, \dots, t_L)$ within the corpus, we first define the complexity based on its preceding context $t_{<i} = (t_1, \dots, t_{i-1})$ as its Shannon Surprisal $S(t_i)$:

$$S(t_i) = -\log_2 P(t_i | t_{<i}; \Theta_T) \quad (1)$$

Simultaneously, we extract the Predictive Entropy $H(t_i)$ to measure the overall uncertainty of the next-token distribution:

$$H(t_i) = -\sum_v P(v | t_{<i}; \Theta_T) \log_2 P(v | t_{<i}; \Theta_T) \quad (2)$$

where v denotes a candidate token in the vocabulary $\mathcal{V}$. These signals capture complementary dimensions of token complexity. Surprisal $S(t_i)$ reflects the localized, post-hoc processing demand of the observed token, whereas predictive entropy $H(t_i)$ models the global, a priori uncertainty across the vocabulary distribution. A higher score indicates a more difficult token and activates more experts.

3.2 Automated Complexity Labeling

To enable the student MoE model to better capture token complexity, we construct a large-scale training set with automatically generated complexity labels. This offline pipeline converts dynamic compute allocation into explicit supervision through three stages:

Batch Inference. We process the raw corpus in batches through a frozen, high-capacity teacher model. Performing solely forward propagation without gradient tracking maximizes throughput, enabling feasible annotation at scale.

Dual-Signal Extraction. Using the teacher’s output logits, we compute the surprisal $S(t_i)$ and predictive entropy $H(t_i)$ at each position i . This process generates two raw signal sequences, $\mathbf{S} = (S(t_1), \dots, S(t_L))$ and $\mathbf{H} = (H(t_1), \dots, H(t_L))$, strictly aligned with the input sequence $\mathbf{x}$. Because the initial token t_1 lacks preceding context, we assign it a zero value for both metrics and exclude it from subsequent normalization.

Robust Normalization. Since raw surprisal and entropy values follow long-tail distributions, standard Min-Max scaling compresses most informative tokens into an indistinguishable narrow range. To mitigate outlier sensitivity, we apply a global robust normalization. For surprisal, we compute corpus-level percentiles to establish bounds $S_{\min}^{\text{robust}}$ and $S_{\max}^{\text{robust}}$, clipping raw scores before scaling:

$$C_S(t_i) = \frac{\text{clip}(S(t_i), S_{\min}^{\text{robust}}, S_{\max}^{\text{robust}}) - S_{\min}^{\text{robust}}}{S_{\max}^{\text{robust}} - S_{\min}^{\text{robust}}} \quad (3)$$

Applying an identical transformation to $H(t_i)$ yields the normalized sequences $\mathbf{C}_S = (C_S(t_1), \dots, C_S(t_L))$ and $\mathbf{C}_H = (C_H(t_1), \dots, C_H(t_L))$. Keeping the targets within $[0, 1]$ reduces the impact of extreme values and makes training more stable.

3.3 Complexity-Aware Routing Mechanism

We propose a Perception-Decision Decoupling strategy for the MoE routing architecture. This design separates continuous complexity estimation from discrete expert allocation, bypassing the gradient optimization issues inherent to discrete routing decisions. Consequently, it stabilizes training while improving controllability.

3.3.1 Recurrent Complexity Memory (RCM) and Perception (CP)

To capture cross-layer dependencies, we integrate a shared Recurrent Complexity Memory (RCM) into the routing architecture. For each layer l , a layer-specific projection processes the hidden state $\mathbf{h}_t^l$, and the RCM updates the contextual state $\mathbf{h}_{l-1}^{\text{rcm}}$:

$$x'_l = \text{Proj}_l(\mathbf{h}_t^l) \quad (4)$$

$$\mathbf{h}_l^{\text{rcm}} = \text{RCM}(x'_l, \mathbf{h}_{l-1}^{\text{rcm}}) \quad (5)$$

A layer-specific two-layer MLP then maps this updated memory state to the dual complexity signals:

$$[\hat{C}_S^l(t), \hat{C}_H^l(t)] = \sigma(\mathbf{W}_2^l \cdot \text{ReLU}(\mathbf{W}_1^l \cdot \mathbf{h}_l^{\text{rcm}})) \quad (6)$$

where $\mathbf{W}_1^l$ and $\mathbf{W}_2^l$ are the trainable weight matrices of the layer- l two-layer MLP. The sigmoid activation σ strictly constrains the outputs to $[0, 1]$, matching the normalized regression targets.

3.3.2 Dynamic Expert Allocator (DEA)

Based on the dual predictions, the DEA determines the expert budget k_t . To map the multi-dimensional signals into a single scalar without introducing new hyperparameters, we aggregate them via a probabilistic union:

$$C_{\text{eff}}(t) = 1 - (1 - \hat{C}_S^l(t))(1 - \hat{C}_H^l(t)) \quad (7)$$

We then compute a tentative count $\tilde{k}_t$ via power-law scaling:

$$\tilde{k}_t = \text{round} (k_{\min} + (C_{\text{eff}}(t))^\gamma \cdot (k_{\max} - k_{\min})) \quad (8)$$

Here, $k_{\min}$ guarantees a performance lower bound, and $k_{\max}$ matches the backbone’s standard Top- k capacity. A bounding operation then finalizes the allocation:

$$k_t = \text{clamp}(\tilde{k}_t, k_{\min}, k_{\max}) \quad (9)$$

Here, γ is a controllable allocation exponent that governs the curvature of the complexity-to-budget mapping:

- **Performance Mode** ($0 < \gamma < 1$): assigns more experts to moderate-complexity tokens, favoring model quality.
- **Balanced Mode** ($\gamma = 1$): applies a linear mapping from effective complexity to expert budget. We use this as the default setting.
- **Efficiency Mode** ($\gamma > 1$): makes allocation more selective, reserving higher expert budgets for tokens with higher complexity.

This design allows for flexible adjustment of inference costs without requiring retraining.

3.4 Predictor Alignment Strategy

To avoid chaotic expert allocation from random initialization, we implement a Predictor Alignment phase. We employ a strict parameter isolation strategy: freezing the entire pre-trained MoE backbone Θ_{MoE} (including self-attention layers, experts, and gates), while exclusively updating the shared RCM Θ_{RCM} and the layer-specific CP predictors Θ_{CP} . Formally, the update rule at step τ

is:

$$\begin{aligned} \Theta_{\text{MoE}}^{(\tau+1)} &= \Theta_{\text{MoE}}^{(\tau)} \\ \Theta_{\text{RCM}}^{(\tau+1)} &= \Theta_{\text{RCM}}^{(\tau)} - \eta \cdot \nabla_{\Theta_{\text{RCM}}} \mathcal{L}_{\text{complexity}} \\ \Theta_{\text{CP}}^{(\tau+1)} &= \Theta_{\text{CP}}^{(\tau)} - \eta \cdot \nabla_{\Theta_{\text{CP}}} \mathcal{L}_{\text{complexity}} \end{aligned} \quad (10)$$

where η is the learning rate. We train the modules via supervised regression to fit the dual complexity signals derived in Section 3.2. The objective minimizes the joint Mean Squared Error (MSE):

$$\mathcal{L}_{\text{complexity}} = \frac{1}{N_{\text{valid}}} \sum_{i=1}^{B \times L} \mathbf{M}_i \cdot \left(\|\hat{C}_S^l(t_i) - C_S(t_i)\|^2 + \|\hat{C}_H^l(t_i) - C_H(t_i)\|^2 \right) \quad (11)$$

where $\mathbf{M}_i$ is a binary mask excluding padding tokens. This isolated optimization equips the model with multidimensional complexity perception, establishing a stable routing initialization without training collapse.

4 Experiments

We conducted extensive experiments and compared them with a series of strong baselines.

4.1 Datasets

For predictor alignment, we construct a mixed corpus of approximately 12K samples. This comprises 10K random sentences from RedPajama (Weber et al., 2024), augmented with 2K correct reasoning chains from GSM8K (Cobbe et al., 2021) and roughly 200 from the MMLU (Hendrycks et al., 2020). To prevent data contamination, we strictly exclude all evaluation splits from the training phase. We report zero-shot accuracy across a diverse benchmark suite (Team, 2024), encompassing MMLU, GSM8K, BBH (Suzgun et al., 2023), ARC-Challenge, ARC-Easy (Clark et al., 2018), Winogrande (Sakaguchi et al., 2020), TruthfulQA (Lin et al., 2022), CommonsenseQA (Talmor et al., 2019), and WMT24++ (Kocmi et al., 2024).

4.2 Settings and Baselines

Settings. We freeze the entire pre-trained MoE backbone (including self-attention layers, experts, and routing gates), exclusively updating the shared

Methods	Active Params	MMLU	GSM8K	ARC-C	ARC-E	Wino	Truth	CSQA	WMT24++	BBH
<i>Static Top-k</i>										
Top-k ($k = 7$)	2.66B	85.53	94.54	91.89	94.43	67.29	74.42	81.84	79.62	58.13
Top-k ($k = 8$)	3B	85.53	95.70	91.99	94.24	68.07	74.42	82.03	80.68	59.26
<i>Dynamic Top-k</i>										
DA-MoE	2.87B	58.33	60.13	72.66	85.94	64.60	14.84	81.81	79.42	42.34
AdaMoE	2.68B	85.51	96.24	91.92	94.23	67.85	74.25	81.90	80.54	58.24
MoE-D ($p = 0.35$)	2.62B	85.75	94.32	91.70	94.34	68.85	73.93	81.84	80.48	57.21
MoE-D ($p = 0.5$)	2.84B	85.97	95.13	92.19	94.53	68.64	73.93	82.42	81.22	58.41
Ada-K	2.67B	86.20	95.47	91.97	94.41	67.95	74.41	82.05	81.12	58.32
<i>Ours</i>										
CAR ($k_{\min} = 6$)	2.68B	86.41	97.27	91.89	94.14	68.97	74.71	82.32	81.21	57.41

Table 1: Main results on the Qwen3-30B-A3B backbone. We report zero-shot accuracy and average activated parameters. The best score on each benchmark is highlighted in bold.

RCM (Θ_{RCM}) and the layer-specific CP predictors (Θ_{CP}). We utilize Qwen3-30B-A3B (128 total experts, activating 8 per token) (Yang et al., 2025) as both the student and its own teacher to extract self-derived complexity labels. For evaluation, we use zero-shot prompting with deterministic generation and fix the evaluation seed to 42 for reproducibility. To alleviate concerns regarding teacher dependency, we also explore the feasibility of using alternative external models as teachers in Appendix B. For predictor alignment, we train for a single epoch using the AdamW optimizer with a learning rate of 1e-3 and a batch size of 4, accumulating gradients every 2 steps to stabilize convergence. We fix the allocation exponent $\gamma = 1$ and evaluate minimum expert bounds $k_{\min} = 6$. Appendix B details the training overhead, computational costs, inference speed, hyperparameter sensitivity analysis and teacher-scale ablations.

Baselines. We compared our approach against Static Top- k ($k \in \{7, 8\}$), which enforces a fixed compute budget, and four representative dynamic allocation strategies: DA-MoE (Aghdam et al., 2024), an attention-driven heuristic allocator; AdaMoE (Zeng et al., 2024), which introduces learnable null experts for adaptive compute skipping; MoE-D (Huang et al., 2024), a confidence-based method using cumulative probability thresholds ($p \in \{0.35, 0.5\}$); and Ada-K (Zhao et al., 2025), an RL-based method that optimizes routing policies via downstream rewards. Detailed configurations are provided in Appendix C. We adopt zero-shot accuracy and the average number of active parameters as our primary evaluation metrics.

4.3 Main Results

Table 1 presents the performance comparison across multiple benchmarks. The Static Top- k rows establish the baseline performance, revealing the limitations of a rigid computational budget. While increasing the active experts from $k = 7$ to $k = 8$ yields marginal performance gains on specific tasks (e.g., GSM8K and WMT24++), it imposes a uniform and high cost on every token, inflating the active parameters from 2.66B to 3B.

For dynamic routing mechanisms, existing baselines exhibit mixed results. Heuristic methods like DA-MoE suffer from severe performance degradation, indicating that raw attention signals are too noisy for stable expert allocation. Although AdaMoE and MoE-D ($p = 0.5$) achieve improvements over the static baseline, their reliance on internal confidence thresholds leads to suboptimal resource allocation on complex tasks. Furthermore, Ada-K, despite leveraging downstream RL optimization, struggles to match the reasoning capacity of optimal allocation, plateauing at 86.20 on MMLU and 95.47 on GSM8K.

In contrast, CAR demonstrates superiority across both reasoning and generation benchmarks while adhering to a strict parameter budget. Operating with only 2.68B active parameters, CAR ($k_{\min} = 6$) significantly outperforms the resource-heavy Static Top-8 on critical tasks, including a notable 1.57 absolute improvement on GSM8K (97.27 vs. 95.70) and a 0.88 improvement on MMLU. This confirms that explicitly tracking token complexity successfully identifies critical reasoning steps and allocates a broader expert ensemble to them. Consequently, CAR maximizes ana-

Methods	MMLU	GSM8K	ARC-C	Truth	N_a
CAR	86.41	97.27	91.89	74.71	2.68B
Random Signal	84.85	93.10	91.24	73.85	2.70B
w/o Surprisal	85.44	94.15	92.19	74.17	2.70B
w/o Entropy	85.92	96.12	91.95	74.52	2.69B
w/o RCM	86.52	96.35	91.68	74.45	2.68B
w/o Norm.	86.38	96.84	91.97	74.65	2.64B
w/o DEA	85.11	94.85	92.28	74.31	2.71B
w/o Align.	86.18	95.33	92.09	74.61	3.00B
E2E Joint	85.09	96.48	91.31	73.32	2.70B

Table 2: Architectural ablation study on the Qwen3-30B-A3B backbone.

lytical capacity while maintaining high inference efficiency.

4.4 Ablation Study

To validate the effectiveness of CAR, we conduct extensive ablation studies, focusing on both architectural components and teacher model scalability.

Architectural Components. We evaluate several variants to assess the contributions of our perception signals and routing strategy: **Random Signal** bypasses the predictor and assigns random complexity scores to test the necessity of difficulty perception; **w/o Surprisal** removes the surprisal signal (relying solely on predictive entropy); **w/o Entropy** removes the predictive entropy signal to evaluate the dual-signal synergy; **w/o RCM** removes the cross-layer Recurrent Complexity Memory, predicting layer-wise complexity independently; **w/o Norm.** replaces robust normalization with standard Min-Max scaling; **w/o DEA** substitutes our continuous power-law mapping with a discrete threshold bucket; **w/o Align.** skips the supervised predictor training, leaving parameters randomly initialized; and **E2E Joint** conducts end-to-end training to optimize the MoE backbone, RCM, and CP predictor simultaneously, bypassing our decoupled alignment.

Results in Table 2 highlight that each CAR component is essential. Relying on a **Random Signal** drastically hurts performance, confirming the necessity of objective difficulty perception. Removing the surprisal-based grounding (**w/o Surprisal**) causes a significant performance drop, proving it is the primary driver of routing accuracy, while **w/o Entropy** validates the dual-signal synergy. Interestingly, **w/o RCM** achieves the highest MMLU score (**86.52**), but suffers a noticeable drop on complex multi-step reasoning tasks like GSM8K. This indicates that while independent

Teacher Model	MMLU	GSM8K	ARC-C	Truth
Qwen3-0.6B	84.82	95.10	90.15	72.85
Qwen3-1.7B	85.76	96.05	91.22	73.90
Qwen3-4B	85.95	96.42	91.45	74.12
Qwen3-14B	86.20	96.95	91.68	74.45
Qwen3-30B-A3B (Ours)	86.41	97.27	91.89	74.71

Table 3: Impact of teacher model size on student performance. The framework maintains robust performance across a wide range of teacher capacities.

Signal Source	MMLU	GSM8K	ARC-C	Truth	N_a
Gating Entropy	85.18	95.20	91.12	74.81	2.55B
Attention Entropy	85.64	96.82	91.47	73.52	2.78B
Dual-Signal (Ours)	86.41	97.27	91.89	74.71	2.68B

Table 4: Impact of different complexity signal sources on routing performance.

layer-wise perception might slightly favor static factual recall, cross-layer memory is crucial for maintaining consistent reasoning trajectories. The **w/o Norm.** and **w/o DEA** gaps confirm the utility of robust scaling and continuous budget mapping. Notably, **E2E Joint** severely degrades reasoning (TruthfulQA drops to 73.32), proving that our decoupled alignment is vital for stabilizing the routing mechanism.

Teacher Model Scalability. To investigate the scalability of the complexity perception signals, we evaluate the performance of CAR when supervised by teacher models of varying scales. While our default configuration utilizes the Qwen3-30B-A3B backbone as a self-teacher, we examine the model’s robustness by generating complexity labels using smaller models ranging from 0.6B to 14B. As shown in Table 3, CAR demonstrates remarkable robustness to teacher capacity. Even when supervised by significantly smaller models (e.g., 0.6B), the student model maintains competitive performance, exhibiting graceful scaling rather than abrupt failure. This suggests that the dual-signal complexity-grounding mechanism effectively extracts meaningful complexity patterns regardless of teacher precision, ensuring that CAR remains deployable in resource-constrained environments by leveraging smaller models.

5 Analysis

5.1 The Validity of the Dual-Signal Objective

We investigate alternative token complexity signals to validate our dual-signal objective. Table 4 compares these internal heuristics against

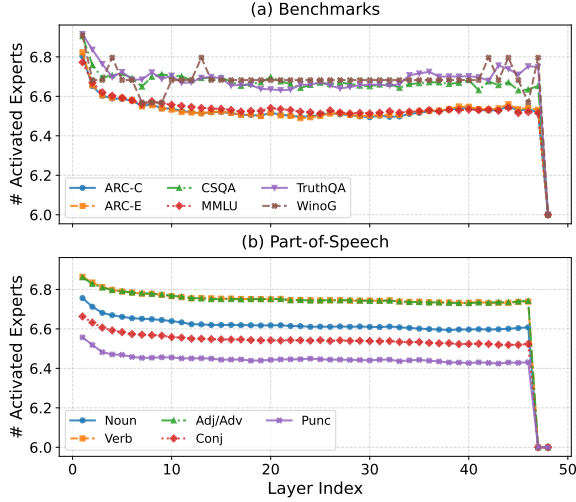


Figure 2: Curves of average expert activations per layer. (a) illustrates distinct activation patterns across benchmarks. (b) decomposes expert activation by Part-of-Speech tags, revealing domain and linguistic-level stratification.

CAR. Gating Entropy enforces the strictest sparsity ($N_a=2.55B$) but suffers severe performance degradation on complex reasoning tasks (e.g., MMLU drops to 85.18 and GSM8K to 95.20). This implies that routing entropy primarily reflects the gate’s optimization ambiguity rather than actual linguistic difficulty. Conversely, Attention Entropy acts as a noisy control signal, incurring the highest computational cost ($N_a=2.78B$) without matching CAR’s accuracy. In contrast, CAR achieves the optimal trade-off by grounding allocation in the explicit dual-signal objective. By decoupling complexity perception from the model’s internal hidden states, CAR ensures that computational resources target intrinsic information density rather than training noise.

5.2 Layer-wise Activation Patterns Analysis

To verify CAR aligns compute with intrinsic token complexity, we analyze the average activated experts across layers ($k_{\min} = 6$), as illustrated in Figure 2. **Domain-Level Stratification.** Figure 2a reveals a compelling correlation between expert usage and benchmark information density. Notably, open-ended knowledge retrieval tasks, such as TruthfulQA and Winogrande, consistently trigger the highest expert activation (peaking around 6.9). In contrast, structured reasoning tasks like ARC-Challenge and MMLU operate more economically (averaging ~ 6.5 experts). This does not imply that structured reasoning is intrinsically easier.

Execution Component	Latency (ms/token)	Proportion
Self-Attention	43.85 ms	1.2%
CP Predictor (Ours)	16.63 ms	0.5%
Dynamic MoE & Memory Access	3455.74 ms	97.3%
Other Overheads	35.10 ms	1.0%
Total End-to-End Latency	3551.32 ms	100.0%

Table 5: Fine-grained inference latency breakdown for CAR on the Qwen3-30B-A3B backbone.

Instead, these benchmarks contain many locally predictable tokens, such as option labels, arithmetic symbols, and formulaic intermediate expressions, which can reduce the average expert budget. Since CAR uses the full effective complexity score rather than surprisal alone, such tokens can still receive higher allocation when predictive entropy or recurrent contextual complexity remains high. **Linguistic-Level Adaptation.** Figure 2b decomposes expert activation by Part-of-Speech tags. We observe that Nouns and Verbs (content words) consistently demand the highest computational budget (maintaining > 6.7 experts in deep layers), reflecting their role as the primary carriers of semantic information. Conversely, Conjunctions and Punctuation trigger significantly fewer experts (< 6.5). This stratification proves that CAR effectively acts as a semantic allocator: it invests heavy compute in information-dense content words while processing predictable syntactic scaffolding with a sparse budget.

5.3 Fine-grained Inference Latency Breakdown

To evaluate the cost-to-benefit ratio, we conduct a micro-benchmark on an NVIDIA A100 GPU to decompose the generation latency (ms/token). As shown in Table 5, the CP predictor introduces only minor overhead (16.63 ms, 0.5%), while the main bottleneck lies in the MoE computation and memory-access phase. The remaining latency gap relative to static Top- k routing mainly comes from variable expert activation, which disrupts the fixed-shape computation and mature CUDA dispatch kernels used by static routing. Our current implementation therefore reflects a practical trade-off between deployability and efficiency; further optimized variable- k dispatch kernels may provide additional speedups and are left for future work.

5.4 Inference Speed Analysis

Table 6 compares the inference throughput of various models. CAR achieves a throughput of 1.47

Methods	Average k	Active Params	Inference Speed (Sample/s)
<i>Dense Models</i>			
Qwen3-4B	-	4B	2.39
Qwen3-8B	-	8B	42.67
<i>Static MoE</i>			
Standard MoE (Top-7)	7.0	2.66B	1.77
Standard MoE (Top-8)	8.0	3B	2.32
<i>Dynamic MoE</i>			
DA-MoE	7.9	2.87B	0.01
AdaMoE	7.1	2.68B	0.70
MoE-D ($p = 0.35$)	6.8	2.62B	0.68
MoE-D ($p = 0.5$)	7.8	2.84B	0.60
Ada-K	7.1	2.67B	0.65
<i>Ours</i>			
CAR ($k_{\min} = 6$)	6.56	2.68B	1.47

Table 6: Comparison of model statistics and inference speed. Inference throughput is tested on the ARC-Challenge dataset using a single NVIDIA A100 GPU.

EP	GPUs	Batch	Avg. k	k CV	k p95	Gen tok/s
No	1	1	6.18	0.063	7.0	1.67
Yes	2	1	6.18	0.063	7.0	1.69
Yes	4	1	6.18	0.063	7.0	2.20
No	1	2	6.25	0.072	7.0	3.73
Yes	2	2	6.25	0.072	7.0	2.55
Yes	4	2	6.25	0.072	7.0	3.59

Table 7: CAR-only expert-parallel throughput. EP denotes expert parallelism; k CV denotes the coefficient of variation of activated expert counts; k p95 denotes the 95th percentile of activated expert counts; Gen tok/s denotes end-to-end generated tokens per second.

samples/s, demonstrating significantly higher efficiency compared to dynamic baselines like Ada-K and MoE-D, while maintaining performance competitive with static baselines. While static Top- k models benefit from highly optimized kernel implementations, CAR narrows the efficiency gap by reducing redundant expert activations, offering a practical balance between computational overhead and reasoning accuracy. Notably, heuristic and RL-based dynamic methods incur higher latency due to complex, iterative decision-making processes, whereas CAR’s decoupled CP predictor operates with minimal sequential overhead, enabling real-time routing adjustments without sacrificing throughput.

5.5 Expert-Parallel Throughput

We further test CAR under expert-parallel deployment with 1, 2, and 4 GPUs. Experts are sharded by expert ID, and routed tokens are dispatched to the corresponding expert devices.

As shown in Table 7, the routing distribution re-

Methods	Active Params	MMLU	ARC-C	Hella	GSM8K	Truth	Wino
Top-6 (Base)	2.80B	44.72	52.22	79.84	16.45	31.07	75.14
AdaMoE	2.72B	45.58	55.35	80.50	15.63	31.48	76.34
MoE-D ($p=0.5$)	2.64B	45.83	55.17	79.24	15.30	32.27	76.24
Ada-K	2.16B	44.98	53.44	80.24	16.86	31.83	75.92
CAR (Ours)	2.31B	46.20	58.50	81.78	16.58	32.08	76.58

Table 8: Performance on DeepSeek-MoE-16B. The best scores are in bold.

mains stable in the tested expert-parallel setting, with Avg. k around 6.18–6.25, small k CV values, and k p95 equal to 7.0. For throughput, CAR reaches 2.20 Gen tok/s with 4 GPUs at batch size 1 and 3.59 Gen tok/s with 4 GPUs at batch size 2. These results do not show severe communication backlogs under the tested setup.

5.6 Results on Other LLMs

We further evaluate our method on DeepSeek-MoE-16B (Dai et al., 2024), guided by DeepSeek-LLM-67B-Chat. As shown in Table 8, CAR achieves the strongest overall performance while maintaining significantly lower expert utilization (2.31B active parameters) than the static Top-6 baseline (2.80B) and strong dynamic methods like AdaMoE (2.72B). Notably, CAR yields substantial improvements on challenging reasoning tasks like ARC-C (+6.28 over static routing) while remaining highly competitive on others like GSM8K. This confirms CAR’s architectural generalizability and its robust ability to dynamically adapt compute to varying task difficulties.

6 Conclusion

In this paper, we presented CAR, a dynamic mechanism designed to address the inefficiencies of static resource allocation in MoE models. By grounding routing decisions in a dual-signal objective—comprising Shannon Surprisal and Predictive Entropy—CAR enables the model to explicitly perceive token complexity and adaptively allocate computational budget. Our decoupled architecture, which separates the Complexity Perception (CP) module (utilizing Recurrent Complexity Memory) from the Dynamic Expert Allocator (DEA), ensures training stability and avoids the collapse issues often observed in heuristic-based dynamic routing. Experimental results demonstrate that CAR consistently outperforms standard Top- k and existing dynamic baselines, particularly on reasoning-intensive benchmarks.

Acknowledgments

The present research was supported by the National Key Research and Development Program of China (Grant No. 2023YFE0116400) and the National Natural Science Foundation of China Youth Fund (Grant No. 62306210). We would like to thank the anonymous reviewers for their insightful comments.

Limitations

While CAR effectively mitigates computational redundancy through objective complexity signals, its empirical scope presents minor constraints for future exploration. First, due to resource limits, our validation focuses on MoE models up to 30B parameters aligned using a restricted corpus of roughly 12K samples. However, consistently outperforming strong baselines under such a sparse data regime strongly underscores the extreme data efficiency and rapid convergence of our decoupled strategy. Second, our framework currently operates as a post-training alignment phase rather than full-parameter pre-training from scratch. Nevertheless, as evidenced by our end-to-end joint training ablations, this decoupled design is a deliberate architectural choice rather than a compromise; it effectively prevents the structural collapse inherent to joint routing optimization, making CAR a highly stable and practical solution for existing massive models.

References

- Maryam Akhavan Aghdam, Hongpeng Jin, and Yanzhao Wu. 2024. Da-moe: Towards dynamic expert allocation for mixture-of-experts models. *arXiv preprint arXiv:2409.06669*.
- Runxi Cheng, Yuchen Guan, Yucheng Ding, Qingguo Hu, Yongxian Wei, Chun Yuan, Yelong Shen, Weizhu Chen, and Yeyun Gong. 2025. Mixture of neuron experts. *arXiv preprint arXiv:2510.05781*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Damai Dai, Chengqi Deng, Chenggang Zhao, Rx Xu, Huazuo Gao, Deli Chen, Jiasli Li, Wangding Zeng, Xingkai Yu, Y Wu, and 1 others. 2024. Deepseek-moe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297.
- Tianyu Dong, Bo Li, Jinsong Liu, Shaolin Zhu, and Deyi Xiong. 2025. Mlas-lora: language-aware parameters detection and lora-based knowledge transfer for multilingual machine translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15645–15660.
- Tianyu Dong, Yangyang Liu, Jiang Zhou, Xinwei Wu, Xiaohu Zhao, Hao Wang, Heng Liu, Linlong Xu, Longyue Wang, Weihua Luo, and 1 others. 2026. Sara: Unlocking multilingual knowledge in mixture-of-experts via semantically anchored routing alignment. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 2830–2846.
- William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.
- Keren Gruteke Klein, Shachar Frenkel, Omer Shubi, and Yevgeni Berzak. 2025. Surprisal takes it all: Eye tracking based cognitive evaluation of text readability measures. *arXiv e-prints*, pages arXiv–2502.
- Yongxin Guo, Zhenglin Cheng, Xiaoying Tang, Zhaopeng Tu, and Tao Lin. 2024. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models. *arXiv preprint arXiv:2405.14297*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024. Harder task needs more experts: Dynamic routing in moe models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12883–12895.
- Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.
- Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

- Peng Jin, Bo Zhu, Yuan Li, and Shuicheng Yan. 2025. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. In *International Conference on Learning Representations*, volume 2025, pages 50832–50856.
- Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, and 1 others. 2024. Findings of the wmt24 general machine translation shared task: The llm era is here but mt is not solved yet. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1–46.
- Dmitry Lepikhin, HyukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.
- Bo Li, Tianyu Dong, Shaolin Zhu, and Deyi Xiong. 2026. Mix-moe: Improving multilingual machine translation of large language models through mixed moes. *IEEE Transactions on Audio, Speech and Language Processing*.
- Shangjie Li, Xiangpeng Wei, Shaolin Zhu, Jun Xie, Baosong Yang, and Deyi Xiong. 2023. Mmmmt: Modularizing multilingual neural machine translation with flexibly assembled moe and dense blocks. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 4978–4990.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.
- Andrey Malinin and Mark Gales. 2020. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*.
- Byung-Doh Oh and William Schuler. 2025. The impact of token granularity on the predictive power of language model surprisal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4150–4162.
- David Raposo, Sam Ritter, Blake Richards, Timothy Lillcrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.
- Zhanming Shen, Hao Chen, Yulei Tang, Shaolin Zhu, Wentao Ye, Xiaomeng Hu, Haobo Wang, Gang Chen, and Junbo Zhao. 2025. Cycle-instruct: Fully seed-free instruction tuning via dual self-training and cycle consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5123–5137.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.
- ModelScope Team. 2024. EvalScope: Evaluation framework for large models.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shrutu Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Maurice Weber, Dan Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, and 1 others. 2024. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37:116462–116492.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Yiheng Yang, Yujie Wang, Chi Ma, Lei Yu, Emanuele Chersoni, and Chu-Ren Huang. 2026. Sparse brains are also adaptive brains: Cognitive-load-aware dynamic activation for llms. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5124–5138.
- Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. 2024. Adamoe: Token-adaptive routing with null experts for mixture-of-experts language models. In *Findings of the Association for*

Computational Linguistics: EMNLP 2024, pages 6223–6235.

Zijia Zhao, Longteng Guo, Jie Cheng, Xuange Gao, Hua Huang, and Jing Liu. 2025. Ada-k routing: Boosting the efficiency of moe-based llms. In *International Conference on Learning Representations*, volume 2025, pages 89619–89635.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, and 1 others. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.

Shaolin Zhu, Leiyu Pan, Dong Jian, and Deyi Xiong. 2025. Overcoming language barriers via machine translation with sparse mixture-of-experts fusion of large language models. *Information Processing & Management*, 62(3):104078.

Shaolin Zhu, Shaoyang Xu, Haoran Sun, Leiyu Pan, Menglong Cui, Jiangcun Du, Renren Jin, António Branco, Deyi Xiong, and 1 others. 2024. Multilingual large language models: A systematic survey. *arXiv preprint arXiv:2411.11072*.

Yuan Zhuang, Yi Shen, Yuexin Bian, Qing Su, Shihao Ji, Yuanyuan Shi, and Fei Miao. 2026. Ld-mole: Learnable dynamic routing for mixture of lora experts. In *International Conference on Learning Representations*, volume 2026, pages 64840–64858.

A Standard Sparse MoE Routing Mechanism

The standard Sparse MoE layer consists of N expert networks $\{E_i\}_{i=1}^N$ and a gating network G . For an input token representation $x \in \mathbb{R}^d$, the gating network computes the routing weights:

$$G(x) = \text{Softmax}(\text{Top-}k(\mathbf{W}_g \cdot x)) \quad (12)$$

where $\mathbf{W}_g \in \mathbb{R}^{N \times d}$ is a trainable weight matrix. The Top- k operation retains the k largest values and sets the rest to $-\infty$. The final output y is the linearly weighted combination of the selected experts' outputs:

$$y = \sum_{i \in \mathcal{T}} G(x)_i \cdot E_i(x) \quad (13)$$

where $\mathcal{T}$ is the set of indices corresponding to the top- k experts. This mechanism ensures that only a small subset of experts is active for any given token, maintaining computational efficiency.

B Appendix Experiments

B.1 More Details About Training Overhead

Table 9 details the architectural specifications of the backbone models employed in our experiments. As shown in Table 10, the training overhead for our approach is relatively efficient. Specifically, the trainable parameters amount to approximately 160M, which is negligible compared to the total 30.6B parameters of the backbone model. This constitutes a mere 0.52% trainable ratio, ensuring that the alignment phase imposes minimal computational cost while effectively adapting the routing mechanism. The training time is less than 3.0 hours on a standard compute node.

B.2 Load Balancing is Maintained

To evaluate the stability of our routing mechanism, we visualize the expert activation frequency distribution on the validation set. Figure 5 compares the expert load distribution of our CAR method (orange bars) against the pre-trained baseline without fine-tuning (blue bars). The horizontal line represents the ideal uniform distribution. We can observe that CAR maintains a healthy and balanced expert utilization profile that closely mirrors the intrinsic distribution of the base model. While there are fluctuations due to the inherent specialization of experts as certain experts naturally handle more

Config	Qwen3-30B-A3B	DeepSeek-MoE-16B
Top- k	8	6
Routed Experts	128	64
MoE Layers	48	27
Activated Params	3B	2.8B
Total Params	30.6B	16.4B

Table 9: Architectural configuration and parameter statistics of the models used in this study. The Qwen3-30B-A3B serves as the primary backbone for our experiments.

Baseline Model	Total Params	Trainable Params	Training Hours
Qwen3-30B-A3B	30.6B	158.7M	3.0h
DeepSeek-MoE-16B	16.4B	83.7M	1.17h

Table 10: Training cost comparison on 12K samples.

frequent linguistic patterns, there are no signs of routing collapse. Most experts remain actively engaged, with activation ratios fluctuating within a reasonable range. This confirms that our framework achieves dynamic compute allocation without sacrificing the load-balancing properties essential for the efficient parallelization and overall capacity of the MoE model.

B.3 Hyperparameter Sensitivity Analysis

The effect of robust bounds $S_{\min}^{\text{robust}}$ and $S_{\max}^{\text{robust}}$. Table 11 compares different clipping thresholds for our robust normalization strategy. We observe that applying a tighter range ($S \in [0.05, 0.95]$) yields superior performance (86.41 on MMLU) compared to the looser $S \in [0.01, 0.99]$ baseline.

This indicates that filtering out the top and bottom 5% of outlier surprisal scores helps the predictor focus on the informative variance of token complexity.

The effect of γ . The allocation exponent γ controls the aggressiveness of expert allocation. As illustrated in Figure 3, setting $\gamma < 1$ (e.g., $\gamma = 0.5$) results in a more generous allocation of experts, prioritizing model performance. Conversely, increasing γ imposes a stricter sparsity constraint. While $\gamma = 1.0$ achieves an optimal balance, pushing γ to 1.2 further reduces the average activated experts (k) to 4.49, offering an ultra-efficient mode. However, this comes at the cost of a noticeable degradation in accuracy (89.65 $\rightarrow$ 89.30). Consequently, we adopt $\gamma = 1$ as the default setting to strike the best trade-off, while re-

Methods	MMLU	ARC-C	ARC-E	Wino	N_a
CAR ($S \in [0.01, 0.99]$)	86.15	91.75	94.10	68.60	2.71B
CAR ($S \in [0.05, 0.95]$)	86.41	91.89	94.14	68.97	2.68B

Table 11: Comparison of different robust normalization ranges. $S \in [a, b]$ denotes the clipping bounds for surprisal scores.

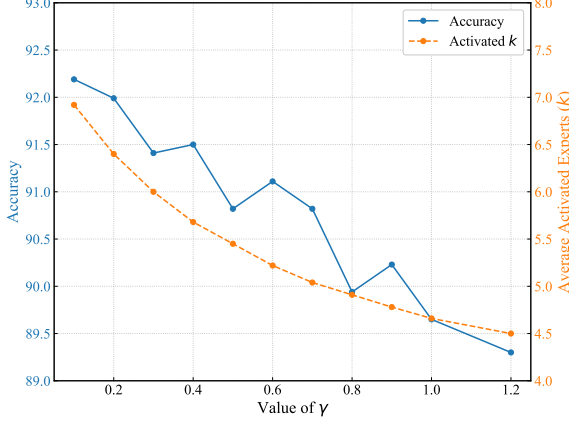


Figure 3: Accuracy and activated k curves on ARC-Challenge at different γ value settings. ($k_{\min} = 4$)

serving $\gamma = 0.5$ as a performance-oriented alternative.

The effect of $k_{\min}$. The parameter $k_{\min}$ sets the minimum computational budget for any token. Figure 4 illustrates that model performance improves significantly as $k_{\min}$ increases from 1 to 6, saturating at $k_{\min} = 7$. This suggests that while dynamic routing is beneficial, a certain baseline capacity is required to maintain robust reasoning capabilities.

B.4 Impact of Training Data Size

Figure 6 investigates the scalability of the predictor training. We observe a logarithmic growth pattern: performance improves rapidly as data increases from 100 to 10,000 samples. Beyond 10k samples, the accuracy gains plateau (91.89 to 91.92 on ARC-C), suggesting that the complexity patterns of language can be effectively learned with a relatively small, high-quality corpus like RedPajama.

B.5 Analysis of Predictor Size and more Complex Structures

We explored the impact of the Complexity Perception (CP) predictor’s capacity. Table 12 shows that increasing the hidden size from 256 to 1024 consistently improves performance across reasoning tasks, with the 1024-dimension predictor achiev-

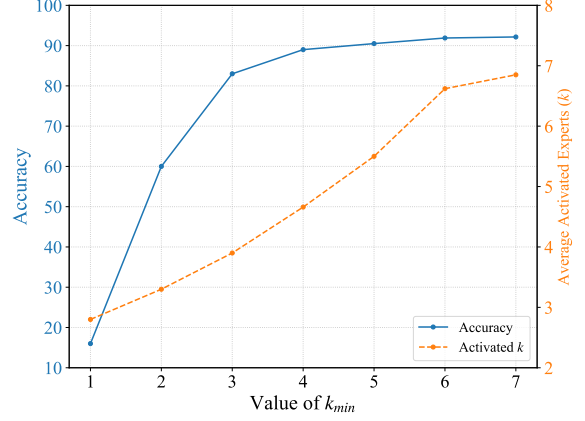


Figure 4: Accuracy and activated k curves on ARC-Challenge at different $k_{\min}$ value settings. ($\gamma = 1$)

ing the highest MMLU score (86.41).

Furthermore, Table 13 compares our standard 2-layer MLP against a Complex MLP variant equipped with RMSNorm, SiLU activation, and Dropout (detailed in Table 14). Surprisingly, the simpler Standard MLP outperforms the complex variant. We hypothesize that the complexity signal is relatively smooth, and the added regularization in the complex variant may hinder the model’s ability to fit the surprisal distribution accurately given the limited training data.

B.6 Ablation on Complexity Signal Source

We analyzed whether to extract the complexity signals (Surprisal and Predictive Entropy) from the teacher’s final output layer or from all intermediate layers. As shown in Table 15 and Figure 7, extracting signals from the last layer results in significantly lower training loss (1.425 vs 1.532) and higher downstream MMLU performance (86.41 vs 83.26). Layer-wise signals introduce noise because intermediate layers typically capture localized syntactic features rather than the global semantic difficulty and overall predictive uncertainty reflected at the final output stage.

B.7 Layer-Wise Uncertainty Dynamics

The activation curves in Figure 2 reflect the evolving complexity of tokens as they propagate through the network. Bottom Layers: We observe uniformly high activation in the initial layers (Layers 0-5), indicating that a broad expert capacity is universally required to encode raw token embeddings into rich latent representations. Middle Layers: Activation stabilizes in the intermediate layers, where the divergence between tasks and POS

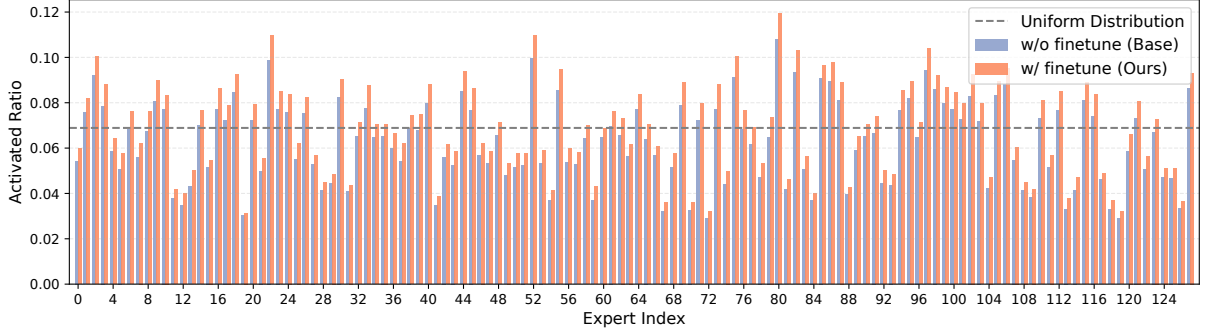


Figure 5: Distribution curves of expert workloads before and after CAR fine-tuning. The gray horizontal line represents the uniform distribution.

Hidden Size	Active Params	MMLU	ARC-C	ARC-E	Wino	Truth	CSQA	AVG
256	2.62B	85.62	91.49	93.84	68.24	74.71	81.54	82.57
512	2.65B	85.86	91.69	94.34	68.65	74.42	81.93	82.82
1024	2.68B	86.41	91.89	94.14	68.97	74.71	82.32	83.07

Table 12: Impact of predictor hidden size on model performance. The highest score on each benchmark is highlighted in bold.

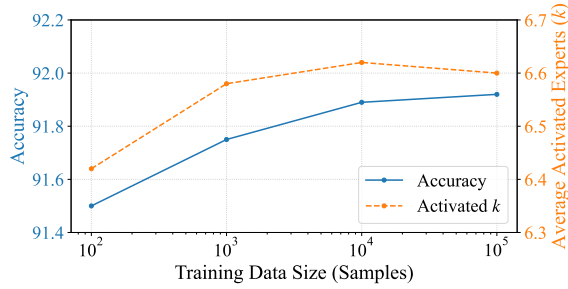


Figure 6: Accuracy score and activated k curves on ARC-Challenge against different sizes of training data.

tags is most pronounced. This confirms that the router adapts to the specific semantic difficulty of the token during the core processing stage. Top Layers: A sharp drop in expert activation is observed in the final layers (Layers 45-48). This aligns with the reduction of predictive uncertainty: as the hidden states converge towards the final logit prediction, the need for diverse expert consultation diminishes, allowing CAR to save significant inference costs.

C Detailed Implementation of Baselines

To ensure a rigorous evaluation, we implemented all baselines within the same Qwen3-30B-A3B model to eliminate implementation-level discrepancies. The specific configurations are as follows:

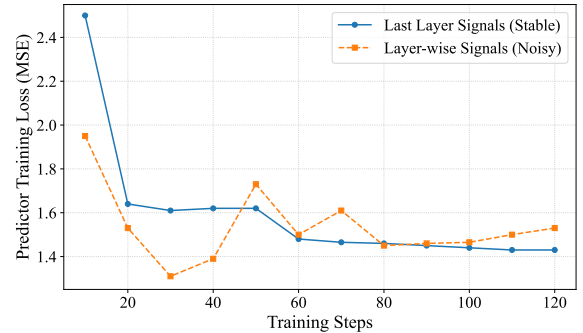


Figure 7: Training loss comparison between using last-layer complexity signals and layer-wise signals. The last-layer extraction demonstrates more stable convergence.

Static Top- k This is the standard routing mechanism native to the Qwen3-MoE architecture. We evaluated two settings:

- Top-8: The standard configuration of the Qwen3-MoE model, which activates eight experts per token to fully utilize the model’s pre-trained capacity.
- Top-7 (Efficiency): We manually reduced k to 7 during inference to establish a slightly more computationally efficient static baseline for comparison.

DA-MoE (Aghdam et al., 2024) DA-MoE (Dy-

Predictor Architecture	Active Params	MMLU	ARC-C	ARC-E	Wino	Truth	CSQA	AVG
Complex MLP	2.67B	85.62	91.40	94.04	68.55	73.81	82.92	82.72
Standard MLP	2.68B	86.41	91.89	94.14	68.97	74.71	82.32	83.07

Table 13: Performance comparison between predictor architectures. Standard MLP consistently outperforms the Complex MLP variant across most benchmarks, demonstrating that a minimal design is sufficient for effective complexity perception.

Component	Standard MLP	Complex MLP
Input Normalization	None	RMSNorm
Projection Layer	Linear (Bias=False)	Linear (Bias=False)
Activation Function	ReLU	SiLU (Swish)
Regularization	None	Dropout ($p = 0.1$)
Output Projection	Linear (Bias=False)	Linear (Bias=False)
Output Range	Sigmoid (0, 1)	Sigmoid (0, 1)

Table 14: Detailed architectural specifications of the complexity predictors. The Complex MLP introduces normalization and regularization mechanisms to theoretically enhance stability, yet empirical results favor the simpler Standard MLP.

Methods	MMLU	Final Loss
Baseline	85.53	-
Layer-wise Signals	83.26	1.532
Last-Layer Signals (Ours)	86.41	1.425

Table 15: Comparison of MMLU performance and final training loss. Using the last-layer dual signals yields lower convergence loss and significantly higher downstream accuracy than the noisy layer-wise signals.

namically Allocated MoE) leverages the Transformer’s self-attention mechanism to quantify token importance and dynamically adjust the computational budget. Unlike heuristic thresholding, DA-MoE computes a specific importance score $S(t)$ for each token by averaging the maximum attention weights across all attention heads. This score is then mapped to the number of active experts k_t via a ceiling function:

$$k_t = \lceil S(t) \times E \rceil \quad (14)$$

where E is the total number of available experts. We implemented this dynamic allocation strategy to strictly follow the original proposal, allowing the model to activate a variable number of experts proportional to the token’s semantic prominence.

MoE-D (Huang et al., 2024) MoE-D employs a confidence-based heuristic. Instead of a fixed k , it selects the smallest set of experts whose cumula-

tive softmax probability exceeds a threshold p .

$$\sum_{i \in \mathcal{T}} G(x)_i \geq p \quad (15)$$

We evaluated two settings reported in the original paper: $p = 0.35$ (efficiency-oriented) and $p = 0.5$ (performance-oriented). To ensure a fair comparison, these thresholds were specifically selected to align the average number of activated parameters with our proposed method. This method requires no training and relies entirely on the pre-trained router’s probability distribution.

AdaMoE (Zeng et al., 2024) AdaMoE modifies the routing space by introducing “null experts” (experts that perform no computation). In our implementation, we configured 16 null experts. Since the original Qwen3 router does not have null experts, we followed the standard adaptation protocol: we fine-tuned the router parameters while keeping the experts frozen. We trained the router on the same 12K mixed samples used for CAR to learn the bias for null expert activation. This ensures a fair comparison where both AdaMoE and CAR are allowed a brief adaptation phase.

Ada-K (Zhao et al., 2025) Ada-K formulates dynamic expert allocation as a reinforcement learning problem, employing a routing policy that dynamically determines k to maximize a reward function balancing performance and efficiency. In our adaptation, to maintain strict fairness, we trained the Ada-K policy using Proximal Policy Optimization (PPO) on the identical 12K mixed alignment corpus. The reward signal was constructed from the target prediction likelihood minus a sparsity penalty. The penalty weight was carefully tuned to yield an average active parameter count of 2.67B, strictly matching the efficiency level of our method. Throughout the training process, all MoE backbone parameters were kept frozen. Different from Ada-K’s reward-driven policy learning, CAR aligns a lightweight complexity predictor using teacher-derived token-complexity

signals. This makes the routing signal less tied to a particular downstream reward design and separates complexity estimation from task-specific policy optimization.

C.1 Fairness of Comparison and Training Efficiency

A potential concern in comparing CAR with heuristic baselines (e.g., MoE-D, DA-MoE) is the introduction of a trainable predictor. We argue that this comparison is fair and valid for three key reasons:

Negligible Parameter Overhead. Unlike methods that require continued pre-training of the MoE experts, CAR adheres to a strict Parameter Isolation strategy. We freeze 99.48% of the model parameters (the entire MoE backbone) and only update the lightweight CP predictor and shared memory (approx. 160M parameters). As shown in Table 10, this results in a training cost of approximately 3.0 GPU hours. This overhead is virtually negligible compared to the inference lifecycle of an LLM, making CAR practically as deployable as training-free heuristics.

Frozen Knowledge Base. Crucially, because the backbone is frozen, CAR cannot learn new factual knowledge or reasoning patterns during the alignment phase. The predictor only learns meta-cognition estimating how difficult the next token is. Therefore, the performance gains observed in Table 1 stem solely from the superior allocation of existing capabilities, not from the injection of new data. This ensures an equitable comparison regarding the model’s knowledge capacity.

Necessity of Learned Signals. The comparison with AdaMoE and Ada-K (which also involve router training) and MoE-D (heuristic-based) reveals a fundamental finding: static heuristics derived from internal confidence or heavy RL optimizations are often suboptimal or unstable proxies for difficulty. The substantial performance gap between CAR and these baselines demonstrates that a small investment in learning a ground-truth complexity signal (Surprisal and Predictive Entropy) yields significantly higher returns than relying on zero-shot heuristics or task-specific reward engineering. Thus, the cost of training the predictor is the necessary price for bridging the gap between internal uncertainty and true linguistic complexity.