INFORMED ROUTING IN LLMS: SMARTER TOKEN-LEVEL COMPUTATION FOR FASTER INFERENCE

Chao Han[1], Yijuan Liang[1,2], Zihao Xuan[3], Daokuan Wu[1], Wei Zhang[1], Xiaoyu Shen[1*]

[1]Institute of Digital Twin, Eastern Institute of Technology, Ningbo
[2]University of Science and Technology of China
[3]The Hong Kong University of Science and Technology
chan@eitech.edu.cn xyshen@eitech.edu.cn

ABSTRACT

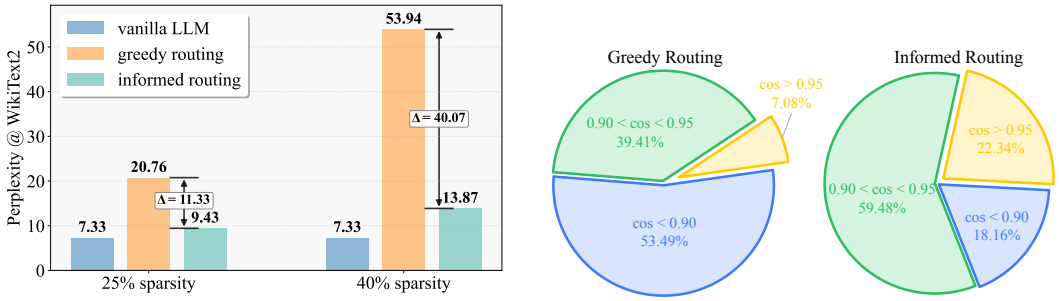
The deployment of large language models (LLMs) in real-world applications is increasingly limited by their high inference cost. While recent advances in dynamic token-level computation allocation attempt to improve efficiency by selectively activating model components per token, existing methods rely on greedy routing—a myopic execute-or-skip mechanism that often leads to irreversible information loss and suboptimal token selection. This paper introduces informed routing, a new paradigm that proactively addresses these issues. The key insight is to assess not only a token's immediate importance but also its recoverability, i.e., how well its transformation can be approximated. To this end, we propose the Lightweight Feature Forecaster (LFF), a small predictive module that estimates a unit's output before routing decisions are made. This enables a flexible execute-or-approximate policy that preserves model fidelity while drastically reducing computation. Extensive experiments on both language modeling and reasoning tasks show that informed routing achieves state-of-the-art efficiency–performance trade-offs across multiple sparsity levels. Notably, even without final LoRA fine-tuning, our method matches or surpasses strong baselines that require full fine-tuning, all while reducing training time by over 50%. The code is available in the GitHub repository: https://github.com/EIT-NLP/informed-routing.

1 INTRODUCTION

The emergence of large language models (LLMs) has catalyzed breakthroughs across diverse industries (Su et al., 2022; OpenAI et al., 2024; Rozière et al., 2024; Cai et al., 2025; Zheng et al., 2025). Scaling laws have established computational requirements as a primary bottleneck in the development and deployment of LLMs (Kaplan et al., 2020; Su et al., 2024). Therefore, reducing this computational overhead has become a key research objective.

Early work primarily focused on **static pruning** methods, which permanently remove a fixed subset of parameters or components from the model (Han et al., 2016; Ma et al., 2023; Xu et al., 2025). While effective for compression, these approaches fail to exploit the varying importance of tokens during inference. More recently, the observation of diverse token criticality has motivated a shift toward **dynamic computation allocation (DCA)** (Raposo et al., 2024; Jiang et al., 2024; Wu et al., 2025; Shin et al., 2025), where different tokens undergo different amounts of computation. DCA partitions the model into computational units—ranging from coarse-grained layers to finer-grained sub-layer components (e.g., self-attention blocks and feed-foward network blocks within a single layer (Zhao et al., 2025)), each equipped with a router. These routers, typically small MLPs, are trained post-hoc to decide whether to execute or skip a unit for each token. In practice, important tokens are routed through most of the model's parameters, while less important ones can skip substantial computation. This flexibility mirrors human language processing, where critical words are analyzed in depth while less informative ones receive only shallow processing.

*Corresponding Author



(a) Perplexity After Router Training

(b) Cosine Similarity Distribution of Features

Figure 1: The limitation of greedy routing and the promise of informed routing. Under the same sparity ratio, informed routing (a) reduces the perplexity and (b) increases feature similarity.

However, existing DCA methods are constrained by a paradigm we term **greedy routing**. Routers are trained to make a simple, binary choice: fully execute a computational unit or skip it entirely. Performance recovery is then attempted via lightweight fine-tuning (e.g., LoRA (Hu et al., 2022)). [1]. The decision to skip is based on minimizing the *immediate* performance drop, without considering the long-term consequences. This greedy approach suffers from two fundamental flaws:

1. **The All-or-Nothing Dilemma:** By forcing a rigid execute-or-skip decision, this paradigm offers no middle ground. Skipping a unit causes irreversible information loss, disrupting the model's internal feature distributions and requiring costly fine-tuning to recover performance. As shown in Figure 1a, skipping tokens leads to a significant increase in perplexity, from 7.33 to 20.76 (25% sparsity) and 53.94 (40% sparsity).
2. **Short-Sighted Token Selection:** The router's focus on immediate impact is a poor proxy for true importance. A token that causes a large immediate drop when skipped is not necessarily indispensable; its transformation might be simple and easily recoverable later. Conversely, a token with low immediate impact might be crucial for maintaining subtle, long-range dependencies that are difficult to restore once lost.

To overcome these limitations, we propose a paradigm shift from greedy decision-making to **informed routing**. Our central idea is to replace the binary execute-or-skip choice with a more nuanced execute-or-approximate decision. We achieve this by equipping each computational unit with a **Lightweight Feature Forecaster (LFF)**—a small, efficient network trained to mimic the output of its larger counterpart. The router's task is no longer to guess which tokens can be safely dropped, but to determine which tokens are *predictable*. If a token's transformation can be accurately approximated by the LFF, it is routed through this efficient path. If the transformation is too complex to be forecasted, the token is processed by the original, powerful unit.

This "informed" approach allows the router to learn a policy based on a token's **recoverability**, not just its immediate impact. For instance, an analysis of feature similarity (Figure 1b) reveals that LFF increases the proportion of features that remain highly similar (cosine similarity > 0.95), from 7.08% to 22.34%. Such significant increase provides concrete evidence for the existence of a substantial fraction of *predictable* tokens—tokens whose transformations can be accurately approximated by the LFF. Based on this insight, we propose a simple three-stage pipeline: (1) train LFF to approximate their corresponding units, (2) train routers to choose between the original unit and its LFF for each token, and (3) perform optional, lightweight LoRA (Hu et al., 2022) fine-tuning to polish the final model. Importantly, we show that we could shrink the original router's intermediate hidden size and reuse the freed capacity to host the LFF, such that *the overall parameter count and computational cost remain identical to standard DCA*, ensuring a fair comparison in both efficiency and resource usage.

Our contributions are as follows:

[1] Some works have also explored jointly training the router and the recovery module (Jiang et al., 2024), however, empirical evidence suggests that this can lead to performance degradation (Zhao et al., 2025).

1. We identify the core limitations of the prevailing **greedy routing** paradigm in DCA, namely its rigid all-or-nothing mechanism and its short-sighted reliance on immediate impact as a routing criterion.
2. We introduce **informed routing**, a new paradigm enabled by **Lightweight Feature Forecaster (LFF)**, which replaces skipping with efficient approximation and allows routing decisions to be based on a token's recoverability.
3. We demonstrate through extensive experiments that our method achieves state-of-the-art efficiency, significantly reducing training overhead and improving final performance. Our analysis further reveals that self-attention modules are highly "predictable", making them prime candidates for approximation.

2 Related Works

Static Pruning Broadly speaking, static pruning techniques related to our approach fall into two main categories: token pruning and parameter pruning. **Static Token Pruning** methods identify and remove tokens deemed redundant, and pruned tokens bypass subsequent transformer layers, which are widely used in Vision-Language Models(VLMs). SpecVLM(Ji et al., 2025) introduces a 'verifier' model to estimate the importance of video tokens. VisionDrop(Xu et al., 2025) identifies token importance via intra-modal attention. In contrast, our dynamic approach achieves flexibility along *model depth*, allowing each token to undergo full computation only in the layers where it is most needed, thereby preserving information while adaptively saving computation. **Static Parameter Pruning** techniques permanently remove fixed structural components(e.g., layers or neurons), resulting in a uniformly smaller model. SliceGPT(Ashkboos et al., 2024) employs Principal Component Analysis (PCA) on the orthogonally transformed parameters, followed by the removal of entire rows/columns. Shortened-llama(Kim et al., 2024) demonstrates that depth pruning is more efficient than width for LLM inference. ShortGPT(Men et al., 2024) proposes Block Influence(BI) to quantitatively estimate the importance of layers in large language models, and subsequently prunes the less important layers. LLM-Streamline(Chen et al., 2025a) removes consecutive layers and then replaces them with a smaller model. Parameter reduction in capacity applies inflexibly to all tokens, regardless of their importance. Conversely, the proposed dynamic method provides flexibility across *input token sequence*, i.e. for a given parameter structure, only a subset of tokens utilizes it while others bypass it via a lightweight path (LFF), enabling a more granular and input-adaptive efficiency.

Dynamic Computation Allocation Dynamic computation allocation methods leverage the observation that linguistic representations evolve at varying paces across tokens. Central to these techniques is the *router* mechanism—a lightweight classifier that dynamically assigns computational paths to tokens. The router acts as a per-token binary classifier at each computation unit. For every token representation, it first computes skip/keep probabilities using a multilayer perceptron. The execution path is then determined via argmax sampling: if "skip" is selected, the token bypasses the unit and remains unchanged; if "keep" is chosen, it undergoes transformation within the unit. This gating mechanism results in dynamic, token-wise computation graphs where inactive tokens propagate directly to the next layer without being processed. Contemporary approaches to dynamic computation allocation exhibit notable variations in sparsity control and computational granularity. Mixture-of-Depths (MoD) (Raposo et al., 2024), enforces a fixed sparsity ratio per layer block, which limits its ability to adapt to input-specific redundancy patterns and ultimately constrains its efficiency. In contrast, subsequent work such as D-LLM (Jiang et al., 2024) introduces global adaptive sparsity, dynamically allocating computation across layers in response to input characteristics. Building on this, SkipGPT (Zhao et al., 2025) further refines the granularity by decoupling attention and MLP operations within each layer. It employs separate routers to independently skip each submodule, enabling more flexible computation paths while maintaining globally optimized sparsity. However, all of these are built with greedy routing.

Error Compensation Prior works have explored error compensation to mitigate compression-induced accuracy drops. RECAP Lee et al. (2025) transfers the statistics of pruned channels to adjacent weights. PRUNE&COMP Chen et al. (2025b) rescales remaining weights offline to compensate for the magnitude gap after layer removal. Olica He & Lin (2025) introduces a linear mapping for low-rank compensation in FFNs. ROE Wu et al. (2025) also identifies the feature gap issue

caused by router skipping in multimodal QA tasks, and implements lightweight network adaptation (termed as adapter) at the instance level. While these methods perform *weight-wise* , *channel-wise* or *instance-wise* compensation, our approach operates in a *token-wise* manner. We dynamically route tokens to either the original model or a LFF, enabling finer-grained and adaptive error recovery. This allows critical tokens to retain full precision while approximating redundant ones, achieving more flexible accuracy-efficiency trade-offs. Additionally, compared with the global alignment of ROE's adapter, our method adopts a local alignment approach for LFF initialization. Specifically, gradients do not need to pass through LLM—this makes the training time of LFF almost negligible compared to the subsequent router/LoRA training, i.e. 5 minutes vs. 3 hours. This aspect constitutes a major contribution of this paper: under 25%/40% sparsity, the proposed informed routing can match or even outperform the performance of greedy routing while saving half of the training time.

3 METHODOLOGY

In this section, we introduce **informed routing**, a framework that replaces rigid skip decisions with efficient approximations. Figure 2 provides an overview of the framework, and we now detail the architecture, the LFF design, and the training procedure.

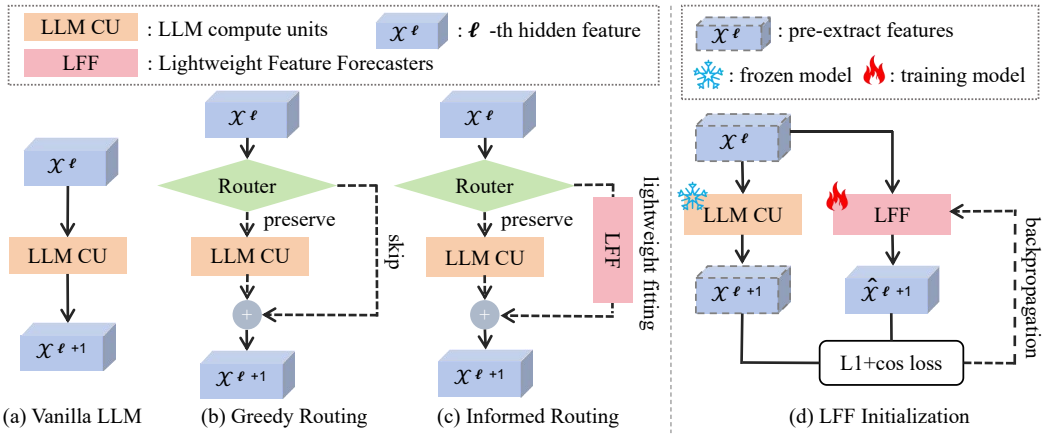


Figure 2: (a), (b), and (c) present the architectural comparison diagrams of the vanilla LLM, greedy routing, and our proposed informed routing paradigm. (d) illustrates how the LFF initialization is performed.

3.1 PRELIMINARIES AND NOTATION

Consider a transformer-based LLM with L layers. For layer $\ell \in \{1, \ldots, L\}$, let $\mathbf{X}^\ell \in \mathbb{R}^{N \times d}$ denote the input token embeddings, where N is sequence length and d is the hidden dimension. Each transformer layer ℓ processes the input through two components, i.e. self-attention and feed-forward network(FFN), with pre-normalization and residual connections:

$$\mathbf{X}_{\text{att}}^\ell = \mathcal{A}^\ell \left(\text{Norm}(\mathbf{X}^\ell)\right) + \mathbf{X}^\ell \quad \text{(Self-attention module)} \tag{1}$$

$$\mathbf{X}^{\ell+1} = \mathcal{F}^\ell \left(\text{Norm}(\mathbf{X}_{\text{att}}^\ell)\right) + \mathbf{X}_{\text{att}}^\ell \quad \text{(FFN module)} \tag{2}$$

where: $\mathcal{A}^\ell$: Multi-head self-attention at layer ℓ, $\mathcal{F}^\ell$: Feed-forward network at layer ℓ, $\mathbf{X}_{\text{att}}^\ell$: Intermediate representation after attention, $\mathbf{X}^{\ell+1}$: Output embeddings serving as input to layer $\ell + 1$.

Following SkipGPT's granularity, we decompose each transformer layer ℓ into two *computational units*: $\mathcal{U}^{\ell,\text{SA}}$ (self-attention) and $\mathcal{U}^{\ell,\text{FFN}}$ (feed-forward network). For each unit, a lightweight *router* $\mathcal{R}^{\ell,k} : \mathbb{R}^d \to \mathbb{R}^2$ (where $k \in \{\text{SA}, \text{FFN}\}$) makes token-wise pruning decisions. The router is implemented as a two-layer MLP with bottleneck dimension, i.e $\mathbb{R}^d \to \mathbb{R}^{\lfloor d/4 \rfloor} \to \mathbb{R}^2$.

The router outputs decision logits for each token $\mathbf{x}_i^{\ell,k}$:

$$\mathbf{r}_i^{\ell,k} = \mathcal{R}^{\ell,k}(\mathbf{x}_i^{\ell,k}) \in \mathbb{R}^2 \tag{3}$$

with routing probabilities obtained via softmax:

$$p_i^{\ell,k} = \sigma(\mathbf{r}_i^{\ell,k})_1 = \frac{\exp(\mathbf{r}_i^{\ell,k}[1])}{\sum_{c=0}^{1} \exp(\mathbf{r}_i^{\ell,k}[c])} \tag{4}$$

where class $c = 1$ indicates *preserving precision through original LLM compute unit* and $c = 0$ indicates *lightweight fitting via LFF branch*. Figure 2(c) illustrates this computational flow.

During the forward pass, a hard binary mask $\mathbf{p}^{\ell,k}$ is sampled by applying the $\arg\max$ operation to Gumbel-Softmax logits, producing discrete $0, 1$ values. In the backward pass, the gradient is estimated using a continuous softmax approximation with temperature τ, enabling differentiable training. The temperature is annealed linearly from $\tau = 5.0$ to $\tau = 1.0$ to sharpen the distribution over time. Modern frameworks such as PyTorch provide built-in functions like $F.gumbel\_softmax$, which facilitates end-to-end training of discrete latent variable models.

The training objective minimizes computation while preserving performance by enforcing a target relative sparsity S_{target} (e.g., 50%). The global computation fraction is:

$$\rho = \frac{1}{2LN} \sum_{\ell=1}^{L} \sum_{k \in \{\text{SA,FFN}\}} \|\mathbf{p}^{\ell,k}\|_0 \tag{5}$$

where $\|\mathbf{p}^{\ell,k}\|_0 = \sum_i p_i^{\ell,k}$, and we regulate ρ toward S_{target} during training (Section 3.3).

3.2 Lightweight Feature Forecaster

The core innovation of our paper is the **lightweight feature forecaster** $\mathcal{F}^{\ell,k} : \mathbb{R}^d \to \mathbb{R}^d$ that approximates the input-output mapping of computational unit $\mathcal{U}^{\ell,k}$ *before* routing decisions. This architectural shift transitions the routing paradigm from reactive recovery to proactive preservation. For efficiency, $\mathcal{F}^{\ell,k}$ uses a bottleneck architecture:

$$\mathcal{F}^{\ell,k}(\mathbf{x}) = \mathbf{W}_2^{\ell,k} \cdot \left(\mathbf{W}_1^{\ell,k}\mathbf{x} + \mathbf{b}_1^{\ell,k}\right) + \mathbf{b}_2^{\ell,k} \tag{6}$$

where $\mathbf{W}_1^{\ell,k} \in \mathbb{R}^{r \times d}$, $\mathbf{W}_2^{\ell,k} \in \mathbb{R}^{d \times r}$ with $r \ll d$ (e.g., $r = 100$ with $d = 4096$ for Llama3.1-8B (Grattafiori et al., 2024)). This yields minimal parameters: $4096 \times 100 + 100 \times 4096 \approx 0.82$M ($0.02\%$ of $\mathcal{U}^{\text{FFN}}$).

$\mathcal{F}^{\ell,k}$ predicts $\mathcal{U}^{\ell,k}$'s normalized output $\mathbf{z}_i^{\ell,k} \triangleq \text{Norm}\left(\mathcal{U}^{\ell,k}(\mathbf{x}_i^{\ell,k})\right)$ using $cosine\ similarity\ loss$ and $L_1\ loss$. Obviously, when the LFF outputs all zeros (i.e., when all its weights are zero), our method degenerates to greedy routing.

3.3 Three-Stage Optimization

Stage 1: LFF Initialization. We commence by training the feature forecasters $\mathcal{F}^{\ell,k}$ to approximate the functional mapping of each computational unit $\mathcal{U}^{\ell,k}$. During this phase, the base LLM parameters remain *frozen*, preserving the original feature distributions. For each unit (ℓ, k), we minimize the forecasting loss $\mathcal{L}_{\text{fit}}^{\ell,k}$ using feature pairs $\{(\mathbf{x}_i^{\ell,k}, \mathbf{z}_i^{\ell,k})\}_{i=1}^N$ extracted from a random-selected subset (2,000 samples) of the training corpus.

This decoupled training paradigm (as shown in Figure 2(d)) admits two significant advantages:

1. *Architectural Independence*: Each $\mathcal{F}^{\ell,k}$ learns a *local approximation* of $\mathcal{U}^{\ell,k}$ without gradient propagation between computational units. This isolation eliminates inter-unit dependencies, enabling:
2. *Massive Parallelization*: Forecasters across all L layers and $k \in \{\text{SA}, \text{FFN}\}$ can be trained concurrently via:

$$\underset{\theta_{\mathcal{F}^{\ell,k}}}{\text{minimize}} \mathbb{F}_{(\mathbf{x},\mathbf{z})\sim\mathcal{D}} \left[\mathcal{L}_{\text{fit}}^{\ell,k}\left(\mathcal{F}^{\ell,k}(\mathbf{x};\theta), \mathbf{z}\right)\right] \quad \forall(\ell,k)$$

where θ denotes forecaster parameters and $\mathcal{D}$ the feature dataset, ensuring computational efficiency during this stage.

Feature tensors $\mathbf{X}^{\ell,k}$ and $\mathbf{Z}^{\ell,k}$ can be precomputed offline, circumventing GPU memory bottlenecks associated with full-model activations. For LLaMA3.1-8B (with 64 LFF), this stage completes in less than 5 minutes on a single NVIDIA RTX 6000 Ada GPU (48GB VRAM).

Stage 2: Router Training. Jointly train routers $\{\mathcal{R}^{\ell,k}\}$ with LLM and forecasters frozen. Router architecture:

$$\mathcal{R}^{\ell,k}(\mathbf{x}) = \text{Linear}_{\lfloor d_1 \rfloor \to 2}\left(\text{ReLU}\left(\text{Linear}_{d \to \lfloor d_1 \rfloor}\left(\mathbf{x}\right)\right)\right) \tag{7}$$

Although the LFF is already sufficiently lightweight, to ensure a fair comparison with baseline methods, we compromise on the parameter configuration of the router to match the parameter count of SkipGPT. Specifically, the intermediate dimension (d_1) of our router is 200 lower than that of SkipGPT. For instance, in the case of the Llama3.1-8B model, SkipGPT adopts an intermediate dimension of $4096/4 = 1024$, while we use $4096/4 - 200 = 824$. As a result, SkipGPT introduces a total of 268.56M parameters (routers), whereas we introduce 268.54M parameters (routers + LFF).

Composite loss integrates:

$$\mathcal{L}_{\text{route}} = \mathcal{L}_{\text{LM}} + \lambda_1 \mathcal{L}_{\text{sparse}} = \mathcal{L}_{\text{LM}} + ||\rho - S_{\text{target}}||_1 \tag{8}$$

where $\mathcal{L}_{\text{LM}}$ is the language modeling loss, ρ is global compute fraction (Eq. 5) and $\lambda_1 = 8.0$ balances two objective.

Stage 3: Parameter-Efficient Fine-tuning. Inject LoRA adapters into attention projections ($\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$) and FFN gates:

$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{AB}, \quad \mathbf{A} \in \mathbb{R}^{d \times r_{\text{LoRA}}}, \mathbf{B} \in \mathbb{R}^{r_{\text{LoRA}} \times d}, \; r_{\text{LoRA}} = 16 \tag{9}$$

Minimize $\mathcal{L}_{\text{LM}}$ with routers/LFF frozen. This step can further recover performance.

4 Experiments

4.1 Experimental Setup

Our experimental configuration aligns with SkipGPT's setting with the following specifications:

Models. We validate the proposed method on the open-source Llama (Grattafiori et al., 2024) model with different scales, i.e. 3B and 8B.

Data. The RedPajama-Data-1T-Sample (Weber et al., 2024) corpus is utilized for both calibration and training.

Evaluation Benchmarks. Performance is assessed on:

- *Reasoning Tasks:* Accuracy on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), ARC-E/C (Clark et al., 2018), and OBQA (Mihaylov et al., 2018) via lm-evaluation-harness (Gao et al., 2024).
- *Perplexity:* Perplexity (PPL) on WikiText-2 (Merity et al., 2017).

Baseline Methods. We selected state-of-the-art static pruning and dynamic computation allocation methods for comparison, with detailed descriptions in section A.3. Please refer to section A.2 in Appendix for detailed training/evaluating settings.

4.2 Results

We conduct extensive experiments to evaluate the proposed informed routing paradigm with LFF. Our results demonstrate its advantages in training stability, efficiency, and performance preservation compared to the traditional greedy routing approaches and static compressing methods. Notably, except for SkipGPT and LFF (ours), all methods report results from LoRA finetuned models. To further demonstrate the effectiveness of our method, we report two-phase results (router training + LoRA finetune) for SkipGPT and LFF, since SkipGPT essentially serves as an ablation experiment for the informed routing component in our method.

Table 1: Performance comparison of different pruning methods on reasoning and language modeling tasks at sparsity levels of 25% and 40%. For reasoning tasks, we report accuracy (%); higher is better. The average (AVG) accuracy across all reasoning tasks is included. For Wikitext-2 (WT2), we report perplexity (PPL); lower is better. The best results under each sparsity level are highlighted in **bold** and the second best are underlined.

(a) Sparsity = 25%

Method	Reasoning (Acc. ↑)								WT2 (PPL ↓)
	BoolQ	OBQA	PIQA	WinoG.	Hella.	ARC-C	ARC-E	AVG	
Dense	82.14	44.6	81.07	77.43	81.89	57.68	84.81	72.80	7.33
Static									
SliceGPT	**72.39**	34.4	66.7	61.56	56.96	31.48	50.08	53.37	9.22
Shortened-llama	71.19	37.4	73.72	**71.82**	69.56	44.45	66.88	62.15	10.32
ShortGPT	72.05	38.4	73.94	70.96	69.23	43.86	68.01	62.35	11.13
Dynamic									
MoD	50.28	31.6	64.25	52.41	50.44	28.24	37.67	44.98	34.21
D-LLM	50.36	30.2	57.4	52.49	37.64	28.16	37.12	41.91	40.12
SkipGPT-Router	54.13	27.6	53.92	54.46	60.92	39.25	68.31	51.23	20.76
LFF-Router (ours)	71.19	40.80	74.97	63.69	73.35	49.23	79.08	64.62	9.43
SkipGPT-Lora	70.67	29.60	56.96	62.83	74.22	49.91	78.79	60.43	10.53
LFF-Lora (ours)	71.93	**41.80**	**76.82**	65.19	**76.54**	**51.45**	**79.38**	**66.16**	**8.91**

(b) Sparsity = 40%

Method	Reasoning (Acc. ↑)								WT2 (PPL ↓)
	BoolQ	OBQA	PIQA	WinoG.	Hella.	ARC-C	ARC-E	AVG	
Dense	82.14	44.6	81.07	77.43	81.89	57.68	84.81	72.80	7.33
Static									
SliceGPT	**67.52**	28.2	60.61	55.41	44.15	25.34	40.7	45.99	14.87
Shortened-llama	65.02	32.4	68.01	64.64	57.55	33.02	53.11	53.39	17.22
ShortGPT	65.38	32.0	68.61	**67.32**	58.43	35.32	53.37	54.35	18.35
Dynamic									
MoD	50.28	33.0	65.56	51.38	54.01	30.2	38.09	46.07	40.42
D-LLM	50.00	31.8	58.54	51.78	48.3	26.88	44.82	44.59	52.78
SkipGPT-Router	53.82	31.8	60.23	54.22	46.02	28.75	52.44	46.75	53.94
LFF-Router (ours)	64.43	36.0	71.87	52.17	59.95	37.71	69.99	56.02	13.87
SkipGPT-LoRA	66.57	37.6	70.78	56.75	65.17	42.66	72.39	58.85	14.35
LFF-LoRA (ours)	65.99	**38.0**	**73.39**	58.8	**69.45**	**43.6**	**72.43**	**60.24**	**11.11**

Inconsistency Between Language Modeling and Reasoning Experimental results reveal an inconsistency between compressed models' language modeling (LM) capability and their reasoning performance. As shown in Table 1, while perplexity (PPL) trends generally align with reasoning accuracy, certain methods deviate. For instance, SliceGPT at 25% sparsity ranks second in PPL but drops to sixth in average reasoning accuracy. Similarly, at 40% sparsity, LFF-Router achieves better PPL than SkipGPT-LoRA yet shows a 3% drop in reasoning. These indicate that LM loss alone may not fully reflect a compressed model's reasoning ability, underscoring the need for evaluation across diverse task-specific benchmarks.

Training Stability and Efficiency Gains of LFF Initialization The proposed informed routing approach significantly improves training stability and efficiency by initializing the router with a pre-fit LFF. This leads to faster and smoother router convergence. Intuitively, after router training, at 25% sparsity, LFF-Router reduces PPL by 11 points compared to SkipGPT-Router; at 40% sparsity, the reduction reaches 40 points. Notably, LFF-Router at 25% sparsity outperforms fully fine-tuned SkipGPT-LoRA while saving over 50% training time (details can be find in section A.2).

Superiority After Fine-tuning and Underlying Mechanisms After LoRA fine-tuning, our method outperforms SkipGPT in 15 out of 16 tasks. We attribute this to two factors: **(1)** The LFF better preserves the original feature distribution by approximating the layer transformation instead of discarding tokens. Features processed by LFF show higher cosine similarity and lower L1 loss (0.16 vs. 0.52), providing a warmer start for fine-tuning. **(2)** Pre-fitting the LFF enables the router to prioritize tokens with high recoverability—those predictable by a simple network—leading to a healthier model structure and better parameter recovery during LoRA fine-tuning.

4.3 Further Analysis

Table 2: Reduction ratio between Attention and MLP modules at different global sparsity levels.

Method	25% Sparsity		40% Sparsity		70% Sparsity	
	Attention	MLP	Attention	MLP	Attention	MLP
SkipGPT	58.0%	42.0%	57.8%	42.2%	56.4%	43.6%
LFF	71.4%	28.6%	67.2%	32.8%	66.2%	33.8%

Analysis of Router Behavior As shown in Table 2, our method consistently select more tokens from self-attention modules than the SkipGPT baseline, across all sparsity levels. This supports prior findings He et al. (2024) that self-attention is more redundant than FFN blocks. The success of our lightweight, linear LFF in predicting attention outputs suggests that many token transformations in self-attention are approximable by simple linear operations. We term this property **linear simplicity**. Our router, preconditioned by the LFF, learns to identify such tokens, leading to a more explainable sparsity profile.

Balanced computation between attention and FFN A potential point of discussion is our treatment of self-attention and FFN modules as equally valid candidates for computation reduction, a design choice inherited from SkipGPT. While self-attention contains fewer parameters, its computational complexity scales quadratically with sequence length, often making it the dominant computational bottleneck in modern long-context large language models. To preemptively address any concern that a direct comparison might be unfair, we conducted a rigorous ablation study. In this experiment, we independently controlled and enforced identical sparsity levels for each module type—self-attention and FFN—across all layers. This ensures a perfectly equitable comparison of the routing strategies' efficiency on a per-module-class basis. The results, showed in Table 3, demonstrate that our informed routing paradigm consistently achieves superior performance compared to the greedy routing baseline under these controlled sparsity conditions. This finding robustly confirms that the performance gain of our method is not an artifact of an imbalanced reduction strategy but is intrinsically linked to its preservation of features distributions, validating our core hypothesis.

Table 3: Performance of SkipGPT-balance and LFF-balance methods under balanced computation reduction setting. All results are reported via LoRA-finetuned model at 25% sparsity.

Method	Reasoning (Acc. ↑)								WT2 (PPL ↓)
	BoolQ	OBQA	PIQA	WinoG.	Hella.	ARC-C	ARC-E	AVG	
SkipGPT-balance	67.77	39.60	62.08	60.93	71.33	40.44	66.84	58.42	11.40
LFF-balance	**70.54**	**39.80**	**74.71**	**63.62**	**73.16**	**47.24**	**75.38**	**63.49**	**9.49**

Table 4: Performance of SkipGPT-LoRA and LFF-Router across different sparsity. Even without the final fine-tuning stage, LFF-Router can already surpass SkipGPT with both router and LLM fine-tuning.

Method	25% Sparsity		40% Sparsity		70% Sparsity	
	Val. Loss↓	PPL ↓	Val. Loss↓	PPL↓	Val. Loss↓	PPL↓
SkipGPT-LoRA	2.38	9.61	2.74	14.34	**3.72**	**52.75**
LFF-Router	**2.36**	**9.46**	**2.57**	**12.89**	4.08	88.17

Can informed routing increase the upper limit of computational reduction? We investigate whether the proposed *informed routing* approach can elevate the upper bound of computational reduction. Empirical results suggest otherwise. As shown in Table 4, at 25% sparsity, LFF-Router *(require only LFF initialization and router training, without LoRA finetuning)* surpasses the full training of SkipGPT-LoRA, indicating effective feature forecasting. However, at 40% sparsity, LFF-Router fails to outperform SkipGPT-LoRA in reasoning tasks, revealing its forecasting limits. At 70% sparsity, the gap widens substantially in language modeling, confirming that LFF's capacity is exceeded. Thus, while effective at moderate sparsity, informed routing does not extend the ultimate computation reduction boundary.

Generalization on Llama-3B To validate the generalization across model scales, we evaluate our method on the Llama3.2-3B model. As shown in Table 5, the results align with those from the 8B

Table 5: Performance comparison on Llama3.2-3B with 25% sparsity. Accuracies (%) on reasoning tasks; perplexity (PPL) on WikiText-2.

Method	Reasoning (Acc. ↑)								WT2 (PPL ↓)
	BoolQ	OBQA	PIQA	WinoG.	Hella.	ARC-C	ARC-E	AVG	
Dense	73.03	43.40	77.58	72.22	76.41	50.85	79.17	67.52	9.27
SkipGPT-Router	47.65	34.40	61.15	54.14	51.99	26.62	39.56	45.07	36.50
LFF-Router (ours)	61.74	34.80	68.28	58.33	62.36	40.53	71.04	56.73	12.19
SkipGPT-LoRA	62.81	36.60	66.49	**59.19**	64.52	39.51	70.71	57.12	14.82
LFF-LoRA (ours)	**62.87**	**37.20**	**69.04**	59.04	**66.14**	**41.81**	**73.06**	**58.45**	**11.46**

model, substantiating our approach's efficacy. At 25% sparsity, LFF-Router significantly outperforms SkipGPT-Router, improving the average accuracy on reasoning tasks by 11% and reducing language modeling perplexity by 24. Moreover, LFF-Router matches the performance of SkipGPT-LoRA while saving over 50% in training time. After LoRA fine-tuning, LFF-LoRA achieves superior performance on 8 out of 9 tasks, confirming the advantage of informed routing. An key finding is that the performance degradation is more pronounced on the 3B model, indicating its lower intrinsic redundancy and higher sensitivity to computation reduction.

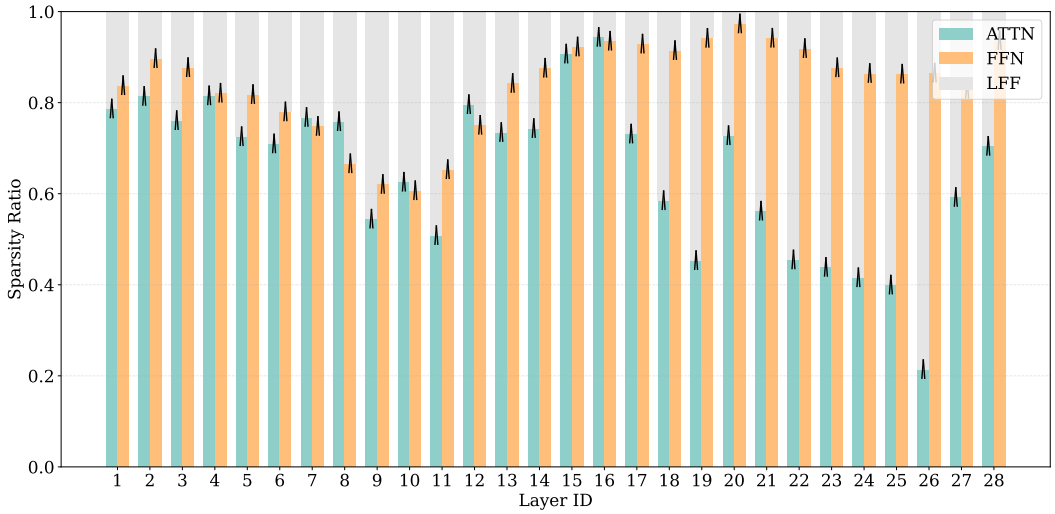


Figure 3: Layer-wise Token Allocation. The hatched area represents the proportion of tokens processed by the efficient LFF branch and the colored areas show the tokens retained for full computation in the Attention (green) and FFN (orange) modules.

Allocation Visualization for Attention and FFN Modules This section presents an intuitive visualization of token allocation by routers across Attention (ATTN) and Feed-Forward Network (FFN) modules within the Llama-3B model. We track and record the layer-wise allocation information for each batch on the validation set, and then present the average value across all batches.

As shown in Figure 3, which details the allocation across 28 layers, the routing mechanism intelligently distributes input tokens, creating a dynamic computational sparsity pattern. A key observation is that the proportion of tokens directed to the LFF branch (hatched area) varies significantly across layers (also between attention and FFN modules), suggesting that the router adapts its filtering strategy based on the hierarchical processing needs of the network. This strategic allocation preserves critical tokens for the full computational pipeline while efficiently processing others, effectively reducing the overall computational overhead without compromising performance.

5 Conclusion

In this work, we identified fundamental limitations in the established greedy routing paradigm for dynamic computation reduction in large language models: its reactive nature leads to irreversible in-

formation loss and its token selection criterion is inherently short-sighted. In response, we proposed a paradigm-shifting alternative, informed routing, which introduces Lightweight Feature Forecasters to fit inter-layer transformations before routing decisions are made. Our approach offers key advantages: First, LFFs approximate skipped tokens, reducing feature shift and improving initial stability with less performance drop. Second, LFF forecasting error gives the router a recoverability-based importance measure, enabling smarter retention of hard-to-predict tokens. Third, our method consistently outperform greedy routing methods on both unbalanced and balanced reduction setting. Finally, we show self-attention's redundancy stems from linearly approximable transformations. Despite these advancements, our exploration of extreme sparsity levels (e.g., 70%) reveals that the upper limit of dynamic computation allocation is ultimately governed by the complexity of the underlying transformations, which a simple LFF cannot fully capture. This presents an exciting avenue for future work, which could explore more sophisticated yet efficient forecasters or hybrid strategies.

REFERENCES

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vXxardq6db.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. Large language models empowered personalized web agents. In *Proceedings of the ACM on Web Conference 2025*, pp. 198–215, 2025.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Representation Learning*, volume 2025, pp. 30362–30383, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4b00a351b41358965613c118e87dc28c-Paper-Conference.pdf.

Xinrui Chen, Hongxing Zhang, Fanyi Zeng, Yongxian Wei, Yizhi Wang, Xitong Ling, Guanghao Li, and Chun Yuan. Prune&comp: Free lunch for layer-pruned llms via iterative pruning with magnitude compensation. *arXiv preprint arXiv:2507.18212*, 2025b.

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

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen

Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*, 2016. URL http://arxiv.org/abs/1510.00149.

Jiujun He and Huazhen Lin. Olica: Efficient structured pruning of large language models without retraining. *arXiv preprint arXiv:2506.08436*, 2025.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. What matters in transformers? not all attention is needed, 2024. URL https://arxiv.org/abs/2406.15786.

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

Yicheng Ji, Jun Zhang, Heming Xia, Jinpeng Chen, Lidan Shou, Gang Chen, and Huan Li. Specvlm: Enhancing speculative decoding of video llms via verifier-guided token pruning. In *The 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://openreview.net/forum?id=mWElG6fKEN.

Yikun Jiang, Huanyu Wang, Lei Xie, Hanbin Zhao, Hui Qian, John Lui, et al. D-llm: A token adaptive computing resource allocation strategy for large language models. *Advances in Neural Information Processing Systems*, 37:1725–1749, 2024.

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, 2020. URL https://arxiv.org/abs/2001.08361.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened LLaMA: A simple depth pruning for large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. URL https://openreview.net/forum?id=18VGxuOdpu.

Mingyu Lee, Akshat Ramachandran, and Tushar Krishna. Recap: Training-free compensation for coarse activation channel pruning in compressed llms. In *Machine Learning for Computer Architecture and Systems*, 2025.

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

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

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect, 2024. URL https://arxiv.org/abs/2403.03853.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, volume abs/1609.07843, 2017. URL https://api.semanticscholar.org/CorpusID:16299141.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan

Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Seungjun Shin, Jaehoon Oh, and Dokwan Oh. Orthorank: Token selection via sink token orthogonality for efficient llm inference. *arXiv preprint arXiv:2507.03865*, 2025.

Hui Su, Xiao Zhou, Houjin Yu, Xiaoyu Shen, Yuwen Chen, Zilin Zhu, Yang Yu, and Jie Zhou. Welm: A well-read pre-trained language model for chinese. *arXiv preprint arXiv:2209.10372*, 2022.

Hui Su, Zhi Tian, Xiaoyu Shen, and Xunliang Cai. Unraveling the mystery of scaling laws: Part i. *arXiv preprint arXiv:2403.06563*, 2024.

Maurice Weber, Dan Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37:116462–116492, 2024.

Qiong Wu, Zhaoxi Ke, Yiyi Zhou, Xiaoshuai Sun, and Rongrong Ji. Routing experts: Learning to route dynamic experts in existing multi-modal large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vtT09dYPGI.

Rui Xu, Yunke Wang, Yong Luo, and Bo Du. Rethinking visual token reduction in lvlms under cross-modal misalignment, 2025. URL https://arxiv.org/abs/2506.22283.

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

Anhao Zhao, Fanghua Ye, Yingqi Fan, Junlong Tong, Jing Xiong, Zhiwei Fei, Hui Su, and Xiaoyu Shen. SkipGPT: Each token is one of a kind. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=d7v2iUSa9s.

Yizhen Zheng, Huan Yee Koh, Jiaxin Ju, Anh T. N. Nguyen, Lauren T. May, Geoffrey I. Webb, and Shirui Pan. Large language models for scientific discovery in molecular property prediction. *Nature Machine Intelligence*, 7(3):437–447, 2025. doi: 10.1038/s42256-025-00994-z.

A Appendix

A.1 Statement on Large Language Model Usage

The authors use Deepseek-r1(DeepSeek-AI et al., 2025) solely for text refinement, including grammar checking, polishing, and condensing sections to meet length constraints.

A.2 EXPERIMENT DETAILS

Training Hyper-parameters differ across stages, all stages adopt the same AdamW optimizer (Loshchilov & Hutter, 2017) ($\beta_1 = 0.9$, $\beta_2 = 0.95$).

- *Forecaster Initialization.* Constant learning rate ($1e^{-3}$), training steps (2000), batchsize (8).
- *Router Tuning:* Constant learning rate (2×10^{-3}) , training steps (2000), batchsize (16).
- *LoRA Tuning:* Cosine annealing learning rate (2×10^{-3}), training steps (2000) with warmup steps (200), batchsize (16).

It is worth emphasizing that in our experiments, we found that computing the router and LFF after normalization (e.g., RMSNorm in llama) can improve training stability—especially when the LFF involves some non-linear activations (e.g., Swish). Therefore, we strongly recommend computing the router and LFF after normalization, which is exactly the approach we adopted in our experiments (to both SkipGPT and LFF).

All experiments were conducted on a single NVIDIA RTX 6000 GPU with 48 GB VRAM. The time consumption of the three experimental phases is summarized in Table 6.

Experimental Phase	Time Consumption
LFF Initialization	5 minutes
Router Training	3 hours
LoRA Finetuning	4 hours

Table 6: Time consumption of different experimental phases.

Evaluating We use lm-eval (Gao et al., 2024) for all evaluation tasks with version 0.4.9. And followed SkipGPT (Zhao et al., 2025), tasks are evaluated with different few-shot contexts, details are listed in Table 7.

Task Name	Number of Few-shot Examples	Evaluation Metric
OpenBookQA	0	acc_norm
Winogrande	5	acc
PIQA	0	acc
HellaSwag	10	acc_norm
BoolQ	0	acc
ARC-Easy	25	acc_norm
ARC-Challenge	25	acc_norm
WikiText2	0	word_perplexity

Table 7: Configuration of few-shot examples and evaluation metrics for different tasks.

A.3 COMPARISON METHODS

To provide a comprehensive evaluation, the proposed method is compared with several state-of-the-art approaches in static model compression and dynamic computation allocation.

- **SliceGPT** (Ashkboos et al., 2024): This method applies Principal Component Analysis (PCA) on orthogonally transformed parameters to remove entire rows and columns, achieving static parameter pruning. It results in a uniformly smaller model by permanently removing fixed structural components.
- **Shortened-llama** (Kim et al., 2024): This approach focuses on depth pruning by removing consecutive layers in LLMs to create a smaller model. It demonstrates that reducing model depth can be an efficient strategy for LLM inference.

- **ShortGPT** (Men et al., 2024): Leveraging Block Influence (BI), ShortGPT quantitatively estimates the importance of layers to prune less critical ones. This method is a static layer-pruning technique that aims to reduce model capacity.
- **Mixture-of-Depths (MoD)** (Raposo et al., 2024): MoD is a dynamic computation allocation method that enforces a fixed sparsity ratio per layer block. It employs a greedy routing paradigm where routers decide to execute or skip computational units for tokens.
- **D-LLM** (Jiang et al., 2024): This method introduces global adaptive sparsity, dynamically allocating computation across layers based on input characteristics. It refines dynamic computation by allowing more flexible computation paths across the model.
- **SkipGPT** (Zhao et al., 2025): A prominent dynamic computation allocation baseline, SkipGPT further refines granularity by decoupling attention and MLP operations within each layer. It uses separate routers to independently skip sub-modules, operating under the greedy routing paradigm.

A.4 Analysis on Key-Value Cache Reduction

While the primary design of SkipGPT and our method focuses on computational reduction, the memory footprint of the Key-Value (KV) Cache remains a critical bottleneck in autoregressive transformer inference. To directly target KV cache reduction, we adopt an aggressive strategy following (Jiang et al., 2024): an additional masking mechanism is applied to prevent normal tokens from attending to any skipped (or route to LFF branch) tokens in the sequence, to simulate that removing the selected tokens' key-value pairs from the cache.

The performance impact of this operation is non-negligible, as it alters the model's fundamental attention pattern. Our experiments (TABLE 8) confirm that enforcing this strict KV cache reduction leads to a predictable degradation in model quality. The training convergence loss increases by 0.19, and the perplexity on WikiText2 rises by 2.3 points compared to the standard pruning setup which retains the full cache. This decline underscores a direct trade-off between memory compression and model fidelity; removing information from the attention context inevitably impairs the model's representational capacity.

Table 8: Performance of models with/without KV reduction at 25% sparsity

Method	Validation Loss	Final PPL ↓
w.o KVR	2.30	8.93
w. KVR	2.49	11.21

A.5 The Selection of Lightweight Feature Forecaster

Table 9: Performance of models with LFFs of different intermediate dimensions at 25% sparsity. Performance saturates beyond a dimension of 50.

Inter. Dim	10	50	100	500	1000
Params. (M: 10^6)	∼0.082	∼0.41	∼0.82	∼4.10	∼8.20
Final PPL ↓	15.9	9.3	8.9	8.9	8.8

The design of the LFF is central to our *informed routing* paradigm. We use a two-layer linear network motivated by two factors: **first**, its simplicity helps validate our core hypothesis—that forecasting before routing stabilizes the training process—without obscuring the gains; **second**, it is highly parameter-efficient, avoiding computational overhead that could undermine acceleration. An ablation on the LFF's intermediate dimension (Table 9) shows performance saturates beyond a dimension of 50, suggesting only a limited subset of token transformations are simple enough to be captured linearly.

Naturally, a more complex forecaster (e.g., an architecture identical to the original model block represents the theoretical upper bound of performance) could achieve better accuracy but offers less speedup, defeating the purpose of inference acceleration. Thus, the LFF lies on a Pareto frontier

between performance and efficiency. Our framework allows users to configure its complexity based on specific accuracy-speed trade-offs, ensuring adaptability across scenarios.