Ruyi2 Technical Report

Huan Song, **Shuyu Tian**, **Junyi Hao**, **Minxiu Xu**, **Hongjun An**, **Yiliang Song**, **Jiawei Shao**, **Xuelong Li**

Institute of Artificial Intelligence (TeleAI), China Telecom

Large Language Models (LLMs) face significant challenges regarding deployment costs and latency, necessitating adaptive computing strategies. Building upon the AI Flow framework, we introduce Ruyi2 as an evolution of our adaptive model series designed for efficient variable-depth computation. While early-exit architectures offer a viable efficiency-performance balance, the Ruyi model and existing methods often struggle with optimization complexity and compatibility with large-scale distributed training. To bridge this gap, Ruyi2 introduces a stable "Familial Model" based on Megatron-LM. By using 3D parallel training, it achieves a 2–3× speedup over Ruyi, while performing comparably to same-sized Qwen3 models. These results confirm that family-based parameter sharing is a highly effective strategy, establishing a new ”Train Once, Deploy Many” paradigm and providing a key reference for balancing architectural efficiency with high-performance capabilities.

Date: February 25, 2026
Model: `https://huggingface.co/TeleAI-AI-Flow/AI-Flow-Ruyi2`
Code: `https://github.com/TeleAI-AI-Flow/AI-Flow-Ruyi2`
Correspondence to: Xuelong Li (`xuelong_li@ieee.org`)

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language understanding and generation tasks (Brown et al., 2020; Bommasani, 2021). However, their rapidly growing parameter scales and inference costs pose significant challenges for practical deployment, particularly in scenarios constrained by latency, throughput, and energy consumption. While Scaling Laws (Kaplan et al., 2020; Hoffmann et al., 2022) indicate that performance improves predictably with increases in model size and data, real-world applications often necessitate “adaptive computing,” where simple inputs do not require invoking the full capacity of the largest model (Schuster et al., 2022; Jiang et al., 2024; Manvi et al., 2024).

Early-exit mechanisms (Teerapittayanon et al., 2016) have emerged as an effective solution to this challenge (Chen et al., 2023b; Elhoushi et al., 2024; Valade, 2024). By incorporating auxiliary output heads within the intermediate layers of transformer models (Xin et al., 2020; Liu et al., 2020), LLMs can terminate inference prematurely upon reaching sufficient confidence, thereby reducing computational costs without the need to retrain multiple independent models. This paradigm effectively transforms a single backbone into a family of nested sub-models with varying depths and computational costs (Elbayad et al., 2019). In this paper, we term such architectures Familial Models and treat them as equivalent to early-exit Large Language Models (Schuster et al., 2022).

Despite its conceptual simplicity, the large-scale training and deployment of Familial Models face multiple challenges. Naive multi-exit training can significantly exacerbate optimization complexity due to gradient interference between different exits across layers. Furthermore, large-scale LLM training typically relies on pipeline, tensor, and data parallelism, and existing early-exit methodologies often exhibit incompatibility with these distributed training mechanisms. Moreover, incorporating instruction tuning into multi-exit settings introduces new questions regarding supervision placement, loss balancing, and parameter efficiency. While approaches such as HELIOS (Kumar et al., 2025) and Balcony (Jamialahmadi et al., 2025) have explored architectural improvements to enhance inference adaptability, and Dynamic Speculative Decoding (Chen et al., 2023a) has been investigated for efficiency, gaps remain. Regarding model scaling, Kim et al. (2024) proposed a simple yet effective depth up-scaling technique (Solar 10.7B) (Kim et al., 2024) to expand LLMs, while LLaMA-Pro (Wu et al., 2024) utilized Block Expansion to efficiently increase capacity. However, most existing methods either focus on small-to-medium scale models or overlook the system-level challenges associated with efficiently training billion-parameter Familial Models. Additionally, while standard Familial Models and SVD-LLM v2 (Wang et al., 2025) have proposed decomposition methods, integrating these into a coherent post-training pipeline remains complex.

Building upon the AI Flow theoretical framework (Shao and Li, 2026; Yuan et al., 2025), we introduce Ruyi2 for efficient variable-depth computation. Although early-exit architectures offer a viable efficiency-performance trade-off, prior implementations such as Ruyi (An et al., 2025) exhibited significant limitations regarding optimization complexity and integration with large-scale distributed systems. To resolve these issues, Ruyi2 implements several critical advancements. Primarily, it introduces a stable Familial Model architecture based on Megatron-LM. By integrating 3D parallel training(Narayanan et al., 2021), the framework is systematically optimized, achieving a 2–3x speedup in training efficiency. Consequently, a single training session yields multiple deployable sub-models, realizing a “Train Once, Deploy Many” paradigm (Cai et al., 2019; Hou et al., 2020). Furthermore, through the rigorous refinement of training data and the application of multi-stage training strategies, the Ruyi2 series consistently outperforms its predecessor, Ruyi, while demonstrating capabilities comparable to same-sized Qwen3 (Yang et al., 2025) models.

This technical report systematically outlines the training of Familial Models, emphasizing instruction tuning, model architecture, and the training framework. We first discuss how to integrate instruction supervision into multi-exit models under both full-parameter and parameter-efficient regimes. Subsequently, we describe the architectural design space of Familial Models, including exit placement, parameter sharing, and head specialization. Finally, we elaborate on the scalable training framework suitable for Familial Models, highlighting how modern distributed systems enable large-scale efficient optimization. To further enhance the capabilities of specific edge-side nodes, we propose the DaE (Expand-then-Decompose) framework for the 1.7B variant. Unlike layer cloning strategies commonly used in previous depth scaling work, we employ Stable Block Expansion (SBE) with random internal initialization to avoid optimization inertia and representation collapse. We then utilize a post-training compression stage derived from SVD-LLM v2 techniques, reducing the parameter count by 40% with only marginal performance degradation.

Experimental results demonstrate that the proposed method achieves excellent performance on reasoning-intensive benchmarks such as GSM8K and MATH. This further verifies the adherence of the familial architecture to scaling laws, proving that the sharing mechanism does not disrupt the backbone’s scaling behavior. This achievement is attributed to the 3D-parallel compatible Familial Models training framework, which resolves the gradient interference challenge of large-scale joint optimization via customized loss aggregation and pipeline-aware backpropagation, enabling full-parameter “zero-overhead” training. For edge-side deployment, this work introduces the DaE paradigm, utilizing zero-residual random initialization to overcome optimization inertia during model expansion, combined with SVD compression to reduce incremental parameters by 40% while keeping performance loss within 2%. Augmented by a multi-stage alignment strategy involving “Continual Pre-training – Fine-tuning – Reinforcement Learning” (incorporating GRPO), this scheme significantly enhances the reasoning capabilities of small-scale sub-models on complex logic tasks, bridging the gap with the backbone model. Ultimately, the system realizes an efficient “Train Once, Produce Many” paradigm, providing a valuable technical pathway for achieving adaptive high-performance inference in resource-constrained environments.

2 Architecture

Consistent with the Ruyi architecture(An et al., 2025), Ruyi2 Familial Models use a *shared backbone* and multiple *exit heads*. The backbone is a standard transformer block. The family-model aspect arises from attaching lightweight output modules at selected depths. For example, one may select layers 6 and 12 to host exit heads. Each exit head ϕ_i takes the hidden state $\mathbf{x}_i$ from layer i and projects it to a token distribution $\mathbf{o}_i$. Ruyi2 Familial Models is designed as follows: Ruyi2-1.7B is a sub-model using layers 1-3, Ruyi2-8B is a sub-model using layers 1-22, and Ruyi2-14B uses the full model with all 40 layers. Notably, all sub-models share the parameters of the common layers, so no separate models need to be trained from scratch. The architecture incorporates a nested early-exit design, where Decomposed Blocks are distributed across various depths of a shared Transformer backbone. This design enables the model to flexibly complete inference and generate Branch Outputs across End devices, Edge servers, or Cloud servers, depending on the available computational resources and task difficulty.

Ruyi2 is built upon Qwen3-14B-Base (Yang et al., 2025), with its largest branch having 14B parameters and supporting early-exit branches with effective parameter sizes of 1.7B and 8B. The 14B main branch was initialized directly from the base model parameters; for the 8B and 1.7B early-exit branches, their decoder layers were initialized using parameters from the layer immediately following their respective early-exit positions. After initialization, we employed a multi-branch joint pre-training approach, conducting continued pre-training on approximately 800 billion tokens from a proprietary high-quality dataset to construct Ruyi2-Base. Subsequently, we performed joint instruction-following fine-tuning on all branches using approximately 4 million high-quality instruction examples, resulting in Ruyi2. Ruyi2 comprises three parameter-sharing models: Ruyi2-14B, Ruyi2-8B, and Ruyi2-1.7B.

Model	Layers	Hidden Size	Tie Embedding	Heads (Q / KV)	Context Length
Ruyi2–1.7B	3	5120	No	40 / 8	8K
Ruyi2–8B	22	5120	No	40 / 8	8K
Ruyi2–14B	40	5120	No	40 / 8	8K

Table 1 Model architecture details for Ruyi2.

3 Pre-training

3.1 Data Acquisition

To build a high-performance foundation model with strong reasoning, coding, and general language capabilities, we constructed a massive, carefully curated pre-training corpus that emphasizes high-density quality over raw web scale. The dataset spans multilingual web text, high-fidelity scientific literature, executable code, and symbolic mathematics. Noisy web data are rigorously filtered and globally deduplicated to retain only information-rich content. Unstructured knowledge from PDFs (e.g., papers, textbooks, manuals) is extracted via a vision-centric pipeline based on fine-tuned vision–language models, followed by a refinement loop using a powerful language model to correct OCR artifacts and restore semantic coherence. Code and math data are sourced from well-structured open-source projects and authoritative academic materials, preserved in original formats (e.g., LaTeX) to maintain structural integrity. Finally, high-quality synthetic data are introduced during mid-training to enhance complex reasoning and general problem-solving abilities.

3.2 Data Processing

Transforming raw data into a training-ready corpus relies on an industrial-grade, multi-stage processing pipeline that prioritizes token quality while preserving semantic diversity. As illustrated in Figure 1, the pipeline integrates multi-level filtering—combining fast heuristic rules with model-based classifiers—to remove low-quality, repetitive, non-target-language, and spam content. To prevent memorization and improve efficiency, we apply advanced multi-granular deduplication, including MinHash/LSH-based fuzzy matching and high-precision suffix-array–based substring deduplication, which is especially effective for code and academic text. Finally, the data mixing strategy is rigorously optimized to establish source weights that balance diversity and complexity. This is coupled with quality-aware upsampling to deliberately increase the exposure of verified, high-value data, enabling higher information density without compromising generalization.

3.3 Pre-training Strategy

3.3.1 Multi-stage Continual Pre-Training Strategy

Our pre-training pipeline employs a dynamic, multi-stage curriculum learning strategy that progressively adjusts data composition and optimizer hyperparameters, allowing the model to evolve smoothly from general

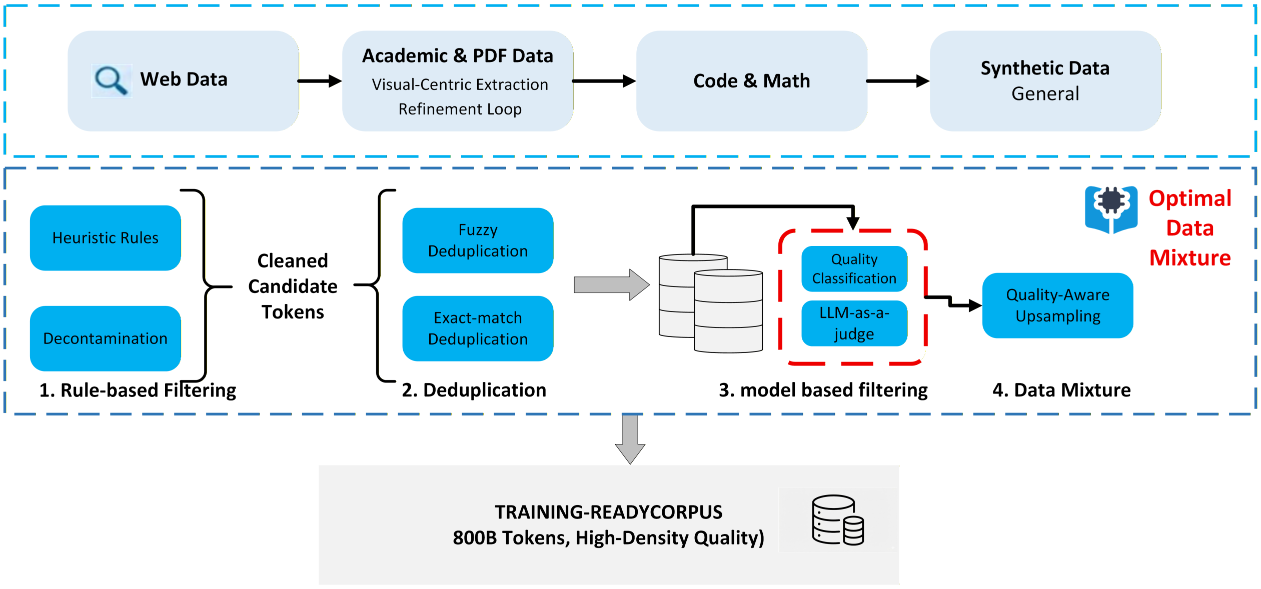


Figure 1 The data engineering pipeline for constructing a high-quality training-ready corpus. The pipeline consists of four main stages: filtering, deduplication, decontamination, and data mixture, ensuring the high density and quality of the dataset.

language modeling to high-level task proficiency. The process begins with General Pre-training, where the model establishes a robust representation space and broad world knowledge by ingesting a large-scale corpus of approximately [NUM] tokens. This provides a solid semantic foundation for the subsequent Mid-training & Annealing phase, which shifts the paradigm toward domain specialization by significantly up-sampling high-quality STEM, code, and high-fidelity synthetic reasoning data. To accommodate this drastic shift in data distribution, we apply a learning rate annealing strategy toward the end of this phase; this smooth decay trajectory mitigates gradient instability and prevents catastrophic forgetting of general representations, ensuring stable convergence and a robust initialization for downstream fine-tuning.

3.3.2 Joint Optimization Framework

We introduce a joint training paradigm in which the main backbone model and all familial branches are optimized simultaneously. This approach ensures that the shared intermediate layers acquire representations that remain robust for both the full-depth model and the early-exiting branches. Formally, given a batch of input tokens, the total training objective $\mathcal{L}_{\text{total}}$ is defined as a weighted sum of the cross-entropy losses from the main backbone and the K auxiliary branches:

$$\mathcal{L}_{\text{total}} = \sum_{k=1}^{K} \lambda_k \mathcal{L}_{\text{branch\_}k} \tag{1}$$

where $\mathcal{L}_{\text{branch\_}k}$ represent the causal language modeling loss for the final layer and the k-th intermediate exit, respectively. The term λ_k denotes a time-dependent weighting coefficient that modulates the influence of auxiliary branches throughout the training process. This scheduling is crucial for providing deep supervision to lower layers in the early stages to accelerate convergence, while progressively decaying effectively to prevent the auxiliary objectives from conflicting with the global optimization of the main backbone during the final convergence phase.

4 Post-training

4.1 Overall Post-training Pipeline

As illustrated in Figure 2, the overall training pipeline introduced in this paper aims to construct a comprehensive family of models, supporting staged capability improvement and coordinated evolution across models of different scales. First, we utilize large-scale domain-specific corpora for *Continued Pre-training (CPT)* to facilitate the transfer from general knowledge to domain knowledge and unify the representation spaces across models of different scales. Second, we adopt a *multi-stage instruction alignment strategy*, which first strengthens fundamental capabilities via general instruction fine-tuning and subsequently introduces domain data to enhance professional performance, ensuring a balance between versatility and specialization. Finally, for small-scale models, we implement an "expand-then-compress" incremental parameter learning strategy: incremental parameters are introduced during training to boost representation capacity and then reduced by 40% using low-rank decomposition before inference. This approach preserves high performance without increasing inference overhead, thereby enabling efficient edge deployment.

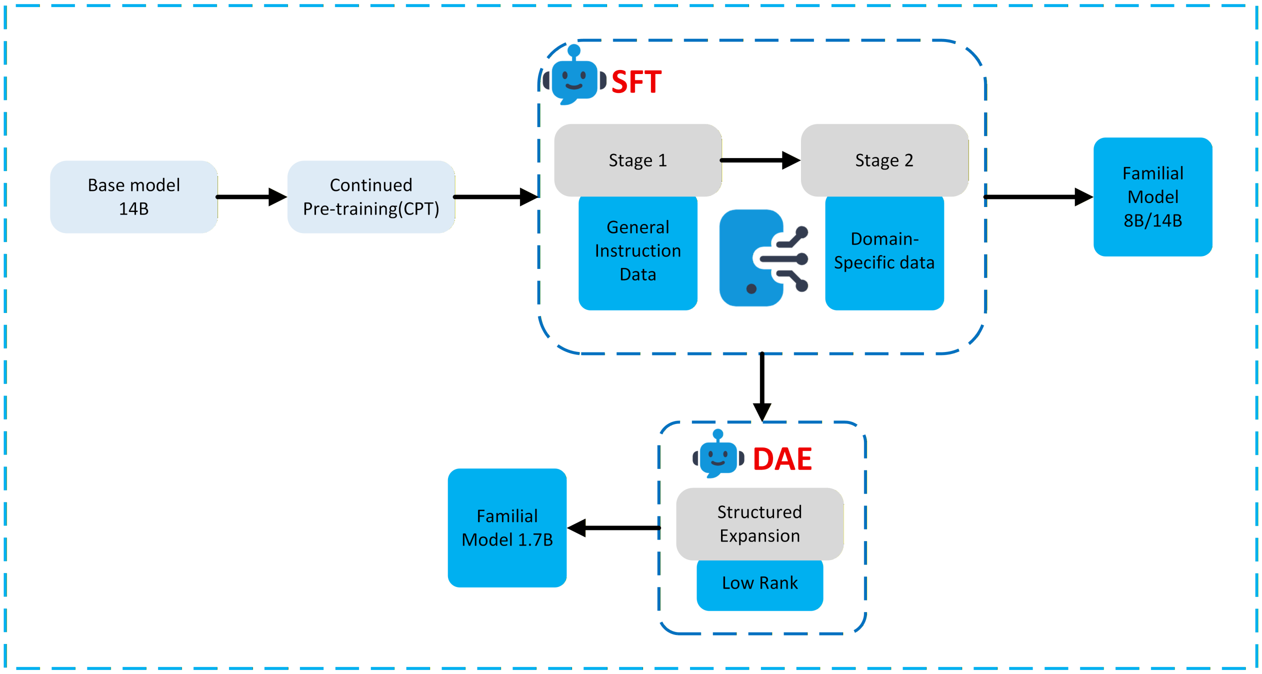
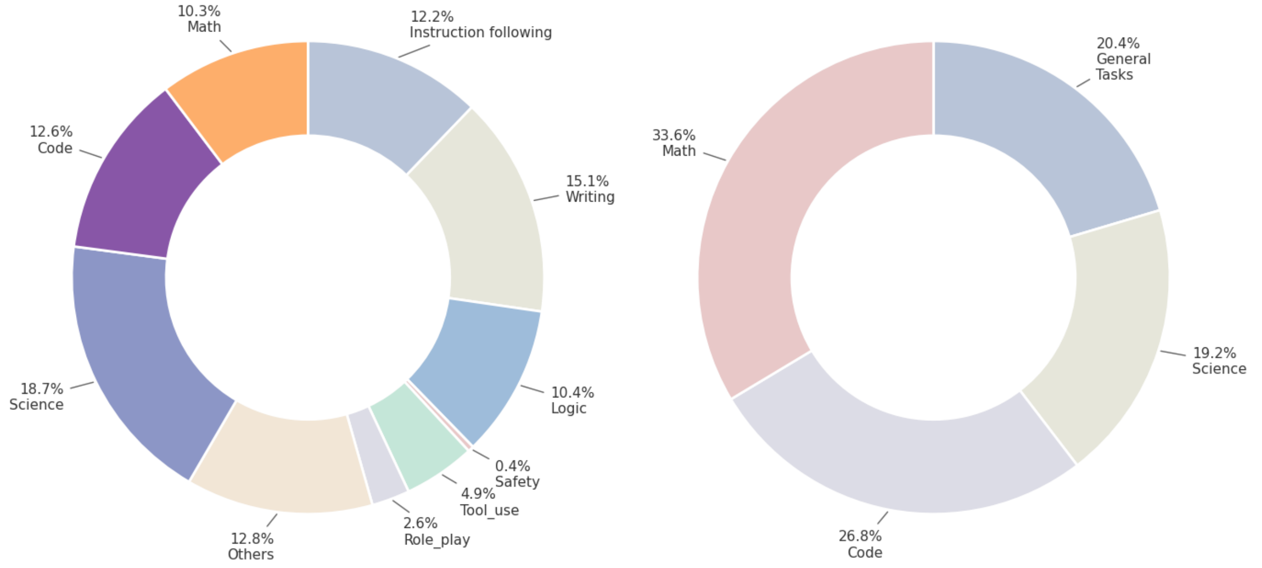


Figure 2 Overview of the Staged Capability Improvement Pipeline

4.2 Post-training Data

Post-training Data During the SFT stage, this study abandons traditional coarse-grained fine-tuning and instead deploys a sophisticated data-engineering system aimed at constructing a high-quality, high-density instruction dataset. The system covers the entire pipeline, including hierarchical categorization, instruction evolution, and multi-dimensional data cleaning. For data curation, we establish a three-level instruction labeling taxonomy spanning over 20 vertical domains. Leveraging high-capability teacher models, we implement Synthetic Instruction Evolution, in which simple instructions are upgraded into high-difficulty composite samples by explicitly introducing complex constraints. Meanwhile, to ensure data purity and safety, we apply a series of stringent filtering and validation procedures, including heuristic rule–based filtering, rigorous decontamination against downstream benchmarks (e.g., GSM8K and HumanEval), and a model-based quality scoring mechamism. For data-mixture decisions, we move away from blind empirical guesswork and adopt a scientific ratio-selection strategy based on proxy models. By conducting extensive ablation studies on small-scale 3B and 7B models, we fit spline curves that characterize the impact of different data categories. This enables the principled determination of optimal mixing ratios. This approach moves from empirical tuning to quantitative optimization. It yields scalable data-mixture strategies that significantly enhance capabilities in mathematics, coding, reasoning, and dialogue. As shown in Figure 3, SFT data construction is divided into two stages: Stage 1 focuses on diversity (Instruction following, Others), while Stage 2 significantly increases the proportion of Math and Code to enhance specific capabilities.



(a) Stage 1: Diverse General Instruction Distribution

(b) Stage 2: Focused Domain-Specific Distribution

Figure 3 SFT Data Mixture Strategy: Evolution from Diverse Generalization to Focused Domain Specialization

4.3 Post-training Strategy

4.3.1 Multi-stage Instruction Alignment

To address the "capability elicitation" bottlenecks inherent in standard supervised fine-tuning (SFT), this study proposes a phased alignment strategy based on curriculum learning principles. Distinct from the knowledge accumulation focus of the pre-training phase, this strategy progressively guides the model from establishing foundational capabilities to mastering specialized domain expertise.

1. **Stage I: General Behavioral Alignment.** This stage aims to build instruction-following capabilities, ensuring consistency and compliance in model interactions. By leveraging diverse data covering multi-turn dialogues and wide-ranging general instructions, the training prioritizes enhancing intent alignment, response precision, and output formatting standards. Simultaneously, it establishes safety boundaries to lay a solid general base for the subsequent expansion of specific task capabilities.

2. **Stage II: Domain Capability Injection.** Building upon general alignment, this stage shifts the optimization focus toward specialized knowledge depth in vertical domains and complex task-solving capabilities. The training data is rigorously curated, encompassing STEM, programming, and tasks with specific constraints. To mitigate the risk of catastrophic forgetting regarding general capabilities during domain adaptation, an Experience Replay mechanism is employed. High-quality general instructions from Stage I are integrated into the training process at a fixed ratio, ensuring the preservation of generalizability while enhancing domain expertise.

4.3.2 1.7B Architecture Specific Optimization

Within our Ruyi2 Familial Models, the 1.7B variant serves as a key node for edge-side deployment, offering a balance between inference latency and generation quality. Nevertheless, due to its reduced depth, this branch has ability constraints in complex semantic comprehension and performing in reasoning tasks when compared to larger familial variants. Addressing these limitations in an efficient and reasonable manner, without training from scratch, remains an important challenge.

Architecture Innovation: The DaE Framework To this end, we propose a two-stage framework termed DaE (Decompose after Expansion), as illustrated in Figure 4, which integrates structured model expansion with post-training compression. In the first stage, inspired by recent advances in efficient scaling methods such as LLaMA-Pro and Solar, we apply a stabilized block expansion strategy to an early-exit branch of the familial model. Specifically, several full-parameter Transformer blocks are appended to the branch, while the original model parameters are frozen and only the newly added blocks are optimized. This design allows the expanded model to fully exploit its representational capacity, thereby achieving enhanced performance prior to subsequent compression.

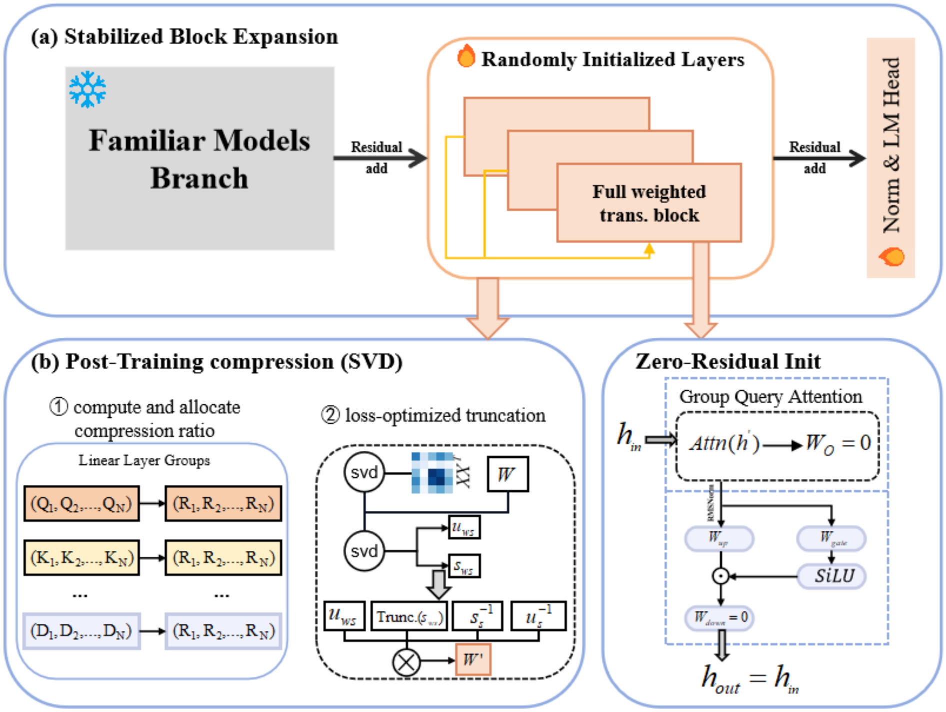


Figure 4 Two stages of DaE

In the second stage, to control the parameter budget and improve deployment efficiency, we apply low-rank decomposition to the newly introduced blocks. Specifically, singular value decomposition (SVD) is employed to re-parameterize the trained full-parameter Transformer blocks into compact low-rank representations. By carefully tuning the compression ratio, this decomposition process substantially reduces both the parameter count and computational overhead, while preserving the majority of performance gains obtained from the block expansion stage. Overall, the proposed two-stage design achieves an effective trade-off between model capacity enhancement and efficiency constraints.

Stabilized Block Expansion Following the methodology of Kim et al., we upscale the depth of the 1.7B branch while preserving its original feature space. Concretely, we append $N = 3$ additional Transformer blocks to the exit of the branch. A central challenge in such block expansion—where newly initialized layers are grafted onto an existing backbone—is maintaining training stability. Naïvely optimizing the expanded model often leads to unstable gradients or representation collapse during the early stages of training.

To address this issue, we adopt a zero-residual initialization strategy. Let h_L denote the hidden state at the exit layer L of the original backbone. For the k-th newly added block ($1 \leq k \leq N$), the forward computation is defined as:

$$h_{L+k} = h_{L+k-1} + \mathrm{Block}_k(h_{L+k-1}), \tag{2}$$

we impose a strict initialization constraint on the final projection matrices of both the attention and MLP submodules:

$$W_o = 0, \quad W_{\mathrm{down}} = 0, \tag{3}$$

where W_o denotes the output projection matrix of the grouped-query attention (GQA) module, and W_{down} represents the down-projection matrix of the MLP. As a result, at initialization time ($t = 0$),

$$\mathrm{Block}_k(h) = 0 \quad \Rightarrow \quad h_{L+k} = h_{L+k-1}, \tag{4}$$

which causes the expanded branch to degenerate into an exact identity mapping of the original exit representation. This initialization scheme preserves the original feature distribution while enabling stable gradient propagation through the newly introduced depths during early training, thereby ensuring a smooth and stable optimization process.

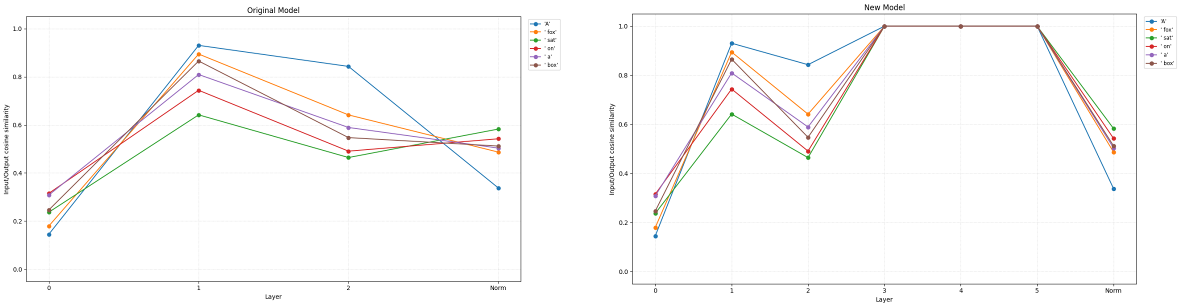


Figure 5 Input/output cosine similarity score of each token per layer for text "A fox sat on a box".

A second critical design choice concerns the initialization of the internal parameters of the newly added blocks. Prior works on depth expansion, such as LLaMA-Pro, commonly adopt layer cloning, where weights from adjacent pretrained layers are duplicated to initialize new blocks. While effective at larger scales, our ablation studies at the 1.7B scale reveal that this strategy introduces pronounced *optimization inertia*: the induced parameter symmetry causes the new blocks to largely replicate existing feature extraction patterns, thereby limiting their capacity to learn novel representations.

To overcome this limitation, we adopt *randomized internal initialization*. Specifically, the internal weights of each new block, namely $(W_Q, W_K, W_V, W_{\mathrm{gate}}, W_{\mathrm{up}})$, are initialized using Gaussian random distributions rather than being cloned from the backbone. When combined with the zero-output constraint described above, this design maximizes the structural entropy of newly added layers while maintaining global stability. Intuitively, random internal initialization prevents representation redundancy, while zero-initialized residual paths reduce the impact of noise on the backbone network. These two mechanisms enable the newly added blocks to anchor their optimization to the original 1.7B model as a stable reference point, while simultaneously expanding the model's functional capacity beyond its original depth. This design effectively decouples capacity expansion from representation perturbation, allowing the model to acquire complementary and higher-order reasoning behaviors without inducing catastrophic interference with existing knowledge structures.

Post-Training Low-Rank Decomposition. While the expanded blocks significantly enhance the model's capacity for complex reasoning, the resulting increase in parameter count introduces substantial challenges for resource-constrained deployment scenarios. In particular, mobile and edge devices typically operate under strict memory and latency constraints, with available system memory often limited to single-digit gigabyte levels after accounting for runtime buffers and auxiliary services. As a result, directly deploying the expanded model in its full-parameter form becomes impractical. To strictly comply with the memory budgets of mobile edge environments (e.g., a ∼4–8 GB RAM budget for model weights and inference buffers), we perform SVD-based weight decomposition on both the newly expanded Transformer blocks and the branch-specific output heads during the second stage of the DaE framework. This compression step is designed to preserve

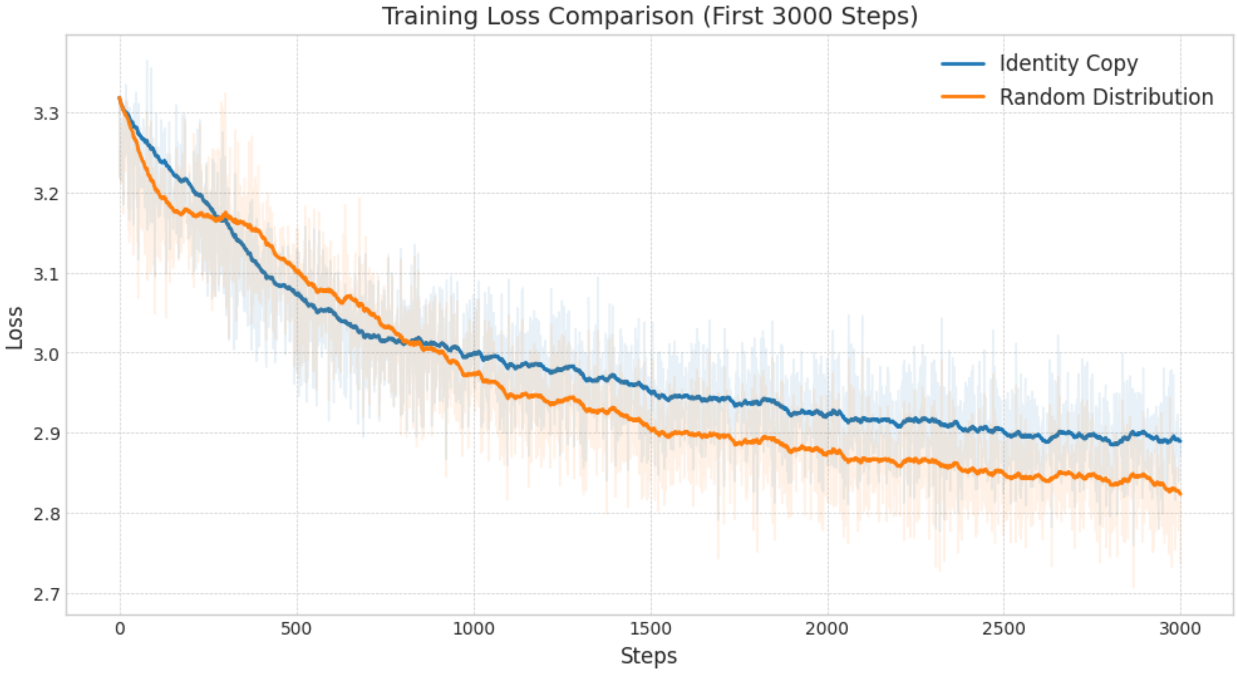


Figure 6 Training loss dynamics on the new model (0-3k steps). The Layer Cloning baseline (red) exhibits optimization inertia with a slower descent rate. In contrast, our Randomized Internal Initialization (purple) consistently achieves lower loss values , demonstrating superior convergence efficiency and a stronger capacity for learning novel representations without being constrained by the backbone's existing patterns.

the majority of the performance gains introduced by depth expansion while substantially reducing memory footprint and computational overhead. Unlike the decomposition described in standard Familial LLMs, our approach is applied as a post-training refinement. We adopt the technique proposed in SVD-LLM v2 to minimize the reconstruction error of layer activations.

(1) **Determining the compression ratio.** We first determine the compression ratio for each layer. A small calibration set is used to compute the input activations $\mathbf{X}$. The theoretical truncation loss is defined as the reconstruction error of the layer output:

$$\mathcal{L}_{\min} = \|\mathbf{WX} - \mathbf{W}'\mathbf{X}\|_F \,, \tag{5}$$

We then apply $1/\log(\mathcal{L}_{\min})$ for inversion and normalization. Finally, the compression ratio for each weight matrix within a group is determined based on the target model compression ratio R and the grouping of weight matrices:

$$\text{Ratio} = \text{Len}(L_G) \times R \times \frac{\mathcal{L}_{\min}}{\text{Sum}(L_G)}, \tag{6}$$

where L_G denotes the list of theoretical truncation losses for all matrices within the same group, $\text{Len}(L_G)$ denotes the group size, and $\text{Sum}(L_G)$ denotes the sum of losses within the group.

(2) **Computing the compressed weight matrix.** We then compute the compressed weight matrix $\mathbf{W}'$. First, from the input activation covariance $\mathbf{X}$, we construct the whitening matrix $\mathbf{S}$ by applying Cholesky decomposition on $\mathbf{XX}^\top$. To ensure numerical stability, we perform SVD on the covariance matrix $\mathbf{XX}^\top$ to obtain $\mathbf{U}_s$, $\mathbf{S}_s$, and $\mathbf{V}_s$. Subsequently, $\mathbf{U}_{ws}$, $\mathbf{S}_{ws}$, and $\mathbf{V}_{ws}$ are obtained by performing SVD on $\mathbf{W} \times \mathbf{U}_s \times \sqrt{\mathbf{S}_s}$. Finally, the decomposed weight matrix is given by:

$$\mathbf{W}' = \mathbf{U}_{ws} \times \text{Trunc.}(\mathbf{S}_{ws}) \times \mathbf{V}_{ws} \times \sqrt{\mathbf{S}_s^{-1}} \times \mathbf{U}_s^{-1}. \tag{7}$$

4.4 Experiment

4.4.1 Experimental Setup

Leveraging the 1.7B branch of our established LLM as the backbone, we implement the SBE strategy by appending N=3 additional Transformer blocks initialized via zero-residual and randomized internal methods; this architecture effectively doubles the total parameter count while maintaining approximately half as trainable parameters. Regarding datasets, we facilitate domain adaptation by pretraining on a 600B-token corpus dominated by STEM and reasoning-intensive texts, supplemented with 10% general replay data to mitigate catastrophic forgetting. Subsequently, the model undergoes Supervised Fine-Tuning (SFT) on roughly 4 million filtered internal entries, followed by a Reinforcement Learning (RL) stage that utilizes filtered mathematical codes and a reserved validation set for reward modeling.

4.4.2 Training Protocol

Distinct from standard fine-tuning paradigms, our training pipeline is designed to strictly adhere to the Parameter-Efficient Expansion principle. Throughout all three stages described below, the parameters of the original 1.7B backbone are completely frozen. Only the newly added expansion blocks and the language modeling head are updated. This constraint ensures that the foundational knowledge of the backbone remains intact while the new layers specialize in higher-order inference.

- **Stage 1: Continued Pre-training (CPT).** The primary objective of this stage is to alleviate the "optimization inertia" of the randomized initialized blocks and align them with the feature space of the frozen backbone. We train the expanded model using a standard autoregressive objective. As demonstrated in our analysis, this stage is crucial for activating the representational capacity of the new layers. We observe that the randomized initialization enables the model to escape the local minima characteristic of layer-cloned weights, achieving a lower perplexity on the validation set.
- **Stage 2: Supervised Fine-Tuning (SFT).** Following CPT, we perform Supervised Fine-Tuning to equip the expanded model with instruction-following capabilities. We configure the training with a global batch size of 128 and a maximum sequence length of 8192 tokens. The model is trained for 3 epochs using a cosine decay learning rate schedule. Specifically, we utilize a linear warm-up of 1000 steps to reach a peak learning rate of 1×10^{-4}, ensuring the newly added parameters adapt smoothly to the instruction distribution without destabilizing the representation.
- **Stage 3: Reinforcement Learning via GRPO.** To further push the boundaries of the model's reasoning capability, we employ Reinforcement Learning (RL) in the final stage. Instead of standard PPO with a separate critic, we adopt Group Relative Policy Optimization (GRPO), which estimates the baseline from the group mean of the outputs.

The stepwise evaluation across distinct training stages demonstrates that the proposed expansion and tuning strategies have a cumulative positive impact on the 1.7B model. Initially, the post-CPT results validate that the added blocks—despite being initialized randomly and trained with a frozen backbone—successfully integrate into the network. This integration provides a verified net gain in capacity rather than introducing noise. Subsequent stages reveal distinct performance benefits. While the introduction of Supervised Fine-Tuning (SFT) provided a substantial boost in instruction following capabilities, Stage 3 (RL) yielded the most significant improvements in logic-intensive tasks, particularly in the MATH benchmark. This suggests that the expanded depth effectively facilitates the complex search capabilities required for multi-step reasoning. Finally, compared to the full-parameter configuration, applying the SVD-based decomposition (Phase 2 of DaE) on the RL-tuned blocks resulted in a marginal performance drop of less than 2%, while significantly reducing the added parameter count by 40%. This indicates that the representations learned via our three-stage protocol maintain high robustness, preserving performance even after substantial model compression.

4.5 Post-training Evaluation

Table 2 demonstrates the superior scaling efficiency and robust performance of the Ruyi2 model family compared to the Qwen3 baselines. Across all parameter scales, Ruyi2 consistently establishes new benchmarks for knowledge understanding and reasoning capabilities.

Unmatched Knowledge Density at 1.7B Even at the compact 1.7B scale, Ruyi2 displays exceptional "knowledge density." It achieves a remarkable MMLU score of 62.77, outperforming Qwen3-1.7B (39.31) by over 23 points. This indicates that Ruyi2-1.7B possesses a foundational knowledge base typically found in much larger models, making it a highly efficient solution for knowledge-intensive tasks despite its small footprint.

Comprehensive Overtake at 8B As the models scale to 8B, Ruyi2 achieves a comprehensive performance overtake. It not only maintains a massive lead in MMLU (79.68 vs. 48.64) but also demonstrates superior reasoning abilities, scoring 92.19 on GSM-8K, significantly higher than Qwen3-8B's 85.37. Consequently, Ruyi2-8B secures a higher overall average score (74.18), validating its architectural superiority in balancing diverse competencies.

Dominance Across All Metrics at 14B The Ruyi2-14B stands out as the premier performer, establishing clear dominance across all key metrics. It extends its lead in general knowledge (MMLU 81.84) and logic (GSM-8K 94.24). Notably, at this scale, Ruyi2 also surpasses Qwen3 in mathematical reasoning (Math 86.52 vs. 84.42). With a commanding average score of 82.88 (vs. 77.77), Ruyi2-14B proves to be the more robust and versatile model for complex, high-level applications.

Model	MMLU	MMLU-P	CMMLU	BBH	ARC-c	Hella	IFEval	Human	Math	GSM8K	Avg.
Qwen3-1.7B	39.31	39.82	61.61	35.81	67.12	54.25	67.65	62.20	70.32	75.89	57.40
Qwen3-8B	48.64	55.63	78.84	55.11	82.03	77.32	81.89	87.20	82.30	85.37	73.43
Qwen3-14B	60.43	64.11	82.06	64.70	81.69	80.80	85.77	87.80	84.42	85.90	77.77
Ruyi2-1.7B	62.77	9.60	22.68	19.95	27.46	58.77	47.13	47.56	39.14	75.97	41.10
Ruyi2-8B	79.68	56.12	74.72	59.38	82.71	78.19	73.94	71.95	72.96	92.19	74.18
Ruyi2-14B	81.84	71.55	82.15	77.86	84.41	83.94	81.52	84.76	86.52	94.24	82.88

Table 2 Performance Comparison between Ruyi2 and Qwen3 Models across benchmarks.

5 Discussion

In this study, the Ruyi2 Familial Models introduces a "one-run, many-models" paradigm that effectively balances architectural efficiency with high-performance capabilities. By integrating early-exit heads into a shared transformer backbone, the model enables adaptive computation where simpler inputs exit early, significantly reducing inference latency without sacrificing the full capacity of the largest variant. The collaborative design of multi-stage instruction tuning and the DaE (Decompose after Expansion) framework establishes a robust pipeline that enhances both general linguistic proficiency and specialized reasoning depth. These advancements provide a scalable, systems-level solution for deploying billion-parameter models under real-world constraints.

5.1 Optimization Complexity and Architectural

Despite the robust performance of the familial architecture, maintaining consistency across nested sub-models poses significant optimization hurdles. Naive multi-exit training often exacerbates optimization complexity, as gradients backpropagated from different exit heads can interfere across the shared intermediate layers, potentially destabilizing the learning process. This interference necessitates careful balancing of loss aggregation to ensure that the shared backbone acquires representations robust enough for both early exits and the full-depth model. Furthermore, during the architectural expansion of the 1.7B variant, we observed that standard techniques like layer cloning introduce pronounced optimization inertia, leading the new blocks to merely replicate existing feature patterns rather than learning novel representations. Consequently, ensuring training stability without succumbing to representation collapse remains a critical systems-level challenge.

5.2 Capability Constraints and Future Directions

Regarding model capabilities, the 1.7B edge variant, while optimized for inference latency, exhibits inherent constraints in complex semantic comprehension and deep reasoning tasks when compared to the larger familial branches. Although the DaE framework helps bridge this gap, the smaller sub-models still face difficulties in achieving the same reasoning depth as the teacher models, particularly in solving complex logic problems without hallucinations. These limitations underscore the ongoing “Open Problems” within the current architecture, specifically regarding long-context processing and the elimination of generative artifacts. Looking forward, future iterations will need to address these gaps by incorporating multimodal integration and exploring extreme sparsification techniques to further enhance the utility of early-exit models in resource-constrained environments.

5.3 Future Evolution Directions and Scaling Law Implications

Future iterations of the Ruyi2 aim to address these limitations while expanding the model’s functional scope. A primary focus will be on resolving persistent open problems common to LLMs, such as mitigating hallucinations and improving stability in long-context processing, which are critical for reliability in complex tasks. Additionally, the research roadmap includes moving beyond the current text-centric paradigm to incorporate multimodal integration, thereby equipping the model with the ability to process and reason across diverse sensory inputs. Finally, to further unlock the potential of edge deployment, future work will investigate techniques for extreme sparsification and the development of more sophisticated dynamic exit policies, aiming to maximize efficiency without compromising the structural integrity of the familial model.

The training and evaluation results of the Ruyi2 situates Familial Models within the broader landscape of Scaling Laws. Our findings suggest that with carefully designed multi-exit loss aggregation, models can retain predictable performance gains associated with increased size while offering the adaptive computation necessary for practical deployment. This indicates that "familial" sharing does not inherently disrupt the fundamental scaling behavior of the underlying transformer backbone.

6 Conclusion

In this work, the Ruyi2 Familial Models is proposed to address the efficiency challenges of large-scale models by enabling adaptive early exits within a shared backbone. The DaE (Decompose after Expansion) strategy is specifically introduced for the 1.7B variant, utilizing stabilized block expansion with randomized internal initialization to enhance reasoning depth, followed by SVD-based decomposition to strictly control parameter size. A systematic multi-stage training pipeline—progressing from continued pre-training to supervised fine-tuning and reinforcement learning via GRPO—is employed to elevate the model’s logical reasoning and instruction-following capabilities. Experimental results show that the enhanced model achieves superior performance on reasoning-intensive benchmarks such as GSM8K and MATH, while maintaining robust general knowledge on MMLU and CMMLU. Notably, the post-training compression reduces the added parameter count by 40% with less than a 2% degradation in performance, validating the robustness of the learned representations. To enable scalable implementation, the system is built on Megatron-LM with full support for 3D parallelism, establishing a "one-run, many-models" paradigm that avoids training separate models from scratch. This architecture effectively handles diverse inference scenarios, offering a flexible solution that balances high-throughput cloud demands with the strict memory constraints of mobile edge devices.

References

Hongjun An, Wenhan Hu, Sida Huang, Siqi Huang, Ruanjun Li, Yuanzhi Liang, Jiawei Shao, Yiliang Song, Zihan Wang, Cheng Yuan, Chi Zhang, Hongyuan Zhang, Wenhao Zhuang, and Xuelong Li. Ai flow: Perspectives, scenarios, and approaches. 2025. URL https://arxiv.org/abs/2506.12479.

Rishi Bommasani. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. *arXiv preprint arXiv:2312.04916*, 2023b.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*, 2019.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, 2024.

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

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

Benyamin Jamialahmadi, Parsa Kavehzadeh, Mehdi Rezagholizadeh, Parsa Farinneya, Hossein Rajabzadeh, Aref Jafari, Boxing Chen, and Marzieh S Tahaei. Balcony: A lightweight approach to dynamic inference of generative language models. *arXiv preprint arXiv:2503.05005*, 2025.

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. *arXiv preprint arXiv:2001.08361*, 2020.

Sanghoon Kim, Dahyun Kim, Chanjun Park, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth upscaling. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 23–35, 2024.

Avinash Kumar, Shashank Nag, Jason Clemons, Lizy John, and Poulami Das. Helios: Adaptive model and early-exit selection for efficient llm inference serving. *arXiv preprint arXiv:2504.10724*, 2025.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6035–6044, 2020.

Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*, 2024.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–15, 2021.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Jiawei Shao and Xuelong Li. Ai flow at the network edge. *IEEE Network*, 40(1):330–336, 2026. doi: 10.1109/MNET.2025.3541208.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

Florian Valade. Accelerating large language model inference with self-supervised early exits. *arXiv preprint arXiv:2407.21082*, 2024.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. Svd-llm v2: Optimizing singular value truncation for large language model compression. *arXiv preprint arXiv:2503.12340*, 2025.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*, 2024.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Cheng Yuan, Zhening Liu, Jiashu Lv, Jiawei Shao, Yufei Jiang, Jun Zhang, and Xuelong Li. Task-oriented feature compression for multimodal understanding via device-edge co-inference. *IEEE Transactions on Mobile Computing*, pages 1–14, 2025. doi: 10.1109/TMC.2025.3626724.