AngelSlim: A more accessible, comprehensive, and efficient toolkit for large model compression

Hunyuan Team

https://github.com/Tencent/AngelSlim
https://huggingface.co/AngelSlim

Abstract

This technical report introduces **AngelSlim**, a comprehensive and versatile toolkit for large model compression developed by the Tencent Hunyuan team. By consolidating cutting-edge algorithms, including quantization, speculative decoding, token pruning, and distillation. AngelSlim provides a unified pipeline that streamlines the transition from model compression to industrial-scale deployment. To facilitate efficient acceleration, we integrate state-of-the-art FP8 and INT8 Post-Training Quantization (PTQ) algorithms alongside pioneering research in ultra-low-bit regimes, featuring **HY-1.8B-int2** as the first industrially viable 2-bit large model. Beyond quantization, we propose a training-aligned speculative decoding framework compatible with multimodal architectures and modern inference engines, achieving $1.8\times$ to $2.0\times$ throughput gains without compromising output correctness. Furthermore, we develop a training-free sparse attention framework that reduces Time-to-First-Token (TTFT) in long-context scenarios by decoupling sparse kernels from model architectures through a hybrid of static patterns and dynamic token selection. For multimodal models, AngelSlim incorporates specialized pruning strategies, namely IDPruner for optimizing vision tokens via Maximal Marginal Relevance and Samp for adaptive audio token merging and pruning. By integrating these compression strategies from low-level implementations, AngelSlim enables algorithm-focused research and tool-assisted deployment.

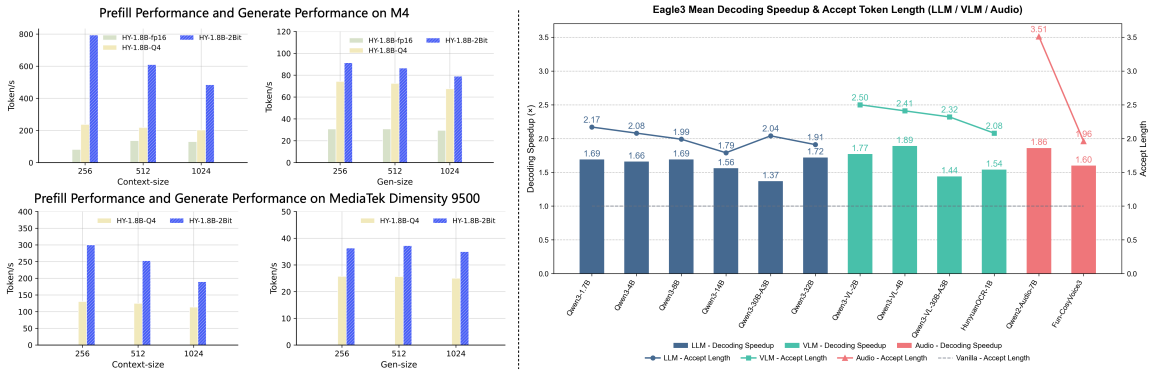

Figure 1: *(Left)* the edge effciency of HY-1.8B-2bit. *(Right)* the speedup of speculative sampling.

Contents

1 Introduction

The landscape of Artificial Intelligence has been fundamentally redefined by the emergence of large-scale foundation models (Achiam et al., 2023; Bai et al., 2023; Wu et al., 2025; Cao et al., 2025; Liu et al., 2024a). Large Language Models (LLMs) (Achiam et al., 2023; Liu et al., 2024a; Bai et al., 2023) and Multimodal Large Models (MLMs) (Wu et al., 2025; Cao et al., 2025) have demonstrated unprecedented capabilities in complex reasoning, zero-shot generalization, and cross-modal synthesis. However, these advancements are inextricably linked to a significant computational paradox known as the "Inference Wall." The deployment of these models is severely hindered by the quadratic scaling of the self-attention mechanism (Vaswani et al., 2017) and the massive memory bandwidth required to fetch billions of parameters. As the demand for longer context windows and real-time multimodal interaction grows, the gap between the hardware requirements of state-of-the-art (SOTA) models and the constraints of commodity GPUs creates a critical bottleneck for the democratization and practical application of AI.

To mitigate these overheads, the research community has proposed several compression paradigms, including quantization (Frantar et al., 2022; Lin et al., 2024), sparsification (Yuan et al., 2025), and speculative decoding (Leviathan et al., 2023). However, existing research typically treats these optimizations as isolated silos. This fragmentation prevents the realization of an end-to-end compression pipeline, as individual methods often fail to account for the cross-technique interference that arises when they are applied concurrently. Consequently, there remains a significant gap between theoretical compression and practical, unified deployment.

In this paper, we present **AngleSlim**, a unified compression and acceleration framework designed to bridge the gap between disparate optimization techniques. Unlike prior works (Frantar et al., 2022; Lin et al., 2024; Yuan et al., 2025) that prioritize a single axis of efficiency, AngleSlim treats quantization, speculative decoding, sparse attention, and token pruning as a cohesive, multi-objective manifold. By synchronizing weight-level precision with dynamic structural sparsity and algorithmic look-ahead, AngleSlim enables large-scale LLMs and MLMs to operate at a fraction of their original computational cost without sacrificing reasoning integrity or multimodal alignment. This holistic integration allows for a hardware-aware optimization process that pushes the boundaries of the inference efficiency frontier.

The first pillar of AngleSlim is a multi-tier quantization suite that addresses the precision-efficiency gap through both Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) (Liu et al., 2025). In the QAT regime, we first incorporate Stretched Elastic Quantization (Liu et al., 2025), an aggressive ultra-low-bit strategy that enables the compression of weights into a 2-bit representation. This allows for the development of highly efficient on-device models, such as our HY-1.8B-2Bit model. Furthermore, we push the boundaries of extreme compression by introducing two novel ternary quantization strategy: **1.58-Bit Tequila** (Huang et al., 2025), a trapping-free ternary quantization method, and **1.25-Bit Sherry** (Huang et al., 2026), which achieves hardware-efficient 3:4 sparse ternary quantization. In the PTQ regime, AngelSlim introduces Leptokurtic Quant (**LeptoQuant**) with Dynamic Outlier Isolation Scale. By observing that weight distributions often follow a Laplacian-like peak with significant outliers, LeptoQuant mitigates the performance degradation typically seen in standard FP8-E4M3 quantization. Instead of allowing traditional smoothing to erode precision in densely populated numerical ranges, LeptoQuant searches for optimal scaling values to isolate outliers and preserve the representational fidelity of the original distribution, ensuring that even the most compressed weights retain high task performance.

Beyond quantization, AngelSlim further targets inference-time efficiency through a speculative sampling–oriented training framework that is explicitly designed to align draft models with their corresponding target models. This framework enables efficient and stable training of Eagle3-style speculative decoders, ensuring high token acceptance rates without sacrificing output correctness. The training pipeline is multimodal, scalable, and production-oriented, supporting language, vision–language, OCR, and speech models, and is directly deployable in modern inference engines such as vLLM. Across a wide range of benchmarks, AngelSlim-trained draft models consistently deliver substantial end-to-end acceleration: Eagle3 (Li et al., 2025a) speculative decoding achieves an average throughput improvement of 1.8–2.0×, while maintaining correctness, with the average number of accepted speculative tokens (AL) ranging from 1.7 to 3.5, depending on model size and modality. These results demonstrate that AngelSlim enables speculative decoding not merely as an inference-time heuristic, but as a training-aligned system capability. Complementary algorithmic advances, such as SpecExit (Yang et al., 2025b), further illustrate the broader design space for dynamic early-exit and adaptive computation, highlighting potential directions for integrating speculative decoding with fine-grained reasoning control in large models.

Moreover, AngelSlim features a training-free sparse attention framework designed to mitigate the computational intensity of long-context inference. By exploiting the inherent sparsity within attention mechanisms, this framework significantly accelerates the prefill process during the KV Cache Generation stage. The system provides a dual-optimization strategy encompassing both static and dynamic regimes. In the static regime, structural heuristics such as A-shape, Tri-shape, and Dilated patterns are employed to maintain a broad receptive field with fixed sparsity masks. In the dynamic regime, AngelSlim integrates advanced algorithms—including MInference (Jiang et al., 2024), XAttention (Xu et al., 2025), FlexPrefill (Lai et al., 2025), and Stem—to selectively compute salient tokens in real-time. By implementing a strict decoupling between sparse kernels and model architectures through a metadata-driven system, the framework offers seamless acceleration for mainstream models like Hunyuan-Dense-Model (Tencent HY Team, 2025d), Llama-3.1 (Dubey et al., 2024), and Qwen3 (Yang et al., 2025a). This approach effectively slashes the Time-to-First-Token (TTFT) while maintaining high accuracy across long-text benchmarks such as LongBench (Bai et al., 2024) and RULER (Hsieh et al., 2024).

Specifically for Multi-modal Large Language Models (MLLMs), visual and audio inputs generate a vast number of tokens, many of which exhibit spatial and temporal redundancy. To address this substantial computational burden, AngelSlim offers token reduction methods for visual and audio modalities, respectively, alongside a unified system framework that enables efficient multimodal acceleration. For the visual modality, we propose IDPruner, which systematically harmonizes token importance and semantic diversity by adapting the Maximal Marginal Relevance (MMR) principle without requiring attention maps. For audio inputs, we introduce Samp, a similarity-attention synergistically driven framework that employs a two-stage merging-pruning pipeline and adaptively calibrates the merging-pruning ratio for each sample. Both methods achieve state-of-the-art performance in their respective modalities and are integrated within a unified system architecture. Besides, by decoupling pruning strategies from model-specific implementations, AngelSlim enables researchers to concentrate on algorithm design without concerning themselves with low-level implementation details. The framework facilitates straightforward deployment with automated benchmarking tools for measuring both task accuracy and inference latency.

In summary, AngelSlim establishes a new SOTA for comprehensive model compression across diverse tasks and benchmarks. Our empirical results demonstrate that the proposed ternary quantization and novel 2-bit QAT framework achieve accuracy parity with INT4 precision while delivering a substantial $4\times$ speedup on edge devices. Furthermore, our high-efficiency 4-8 bit PTQ suite ensures near-lossless compression with a low-memory footprint, while the Eagle3 framework provides a seamless, training-aligned pipeline for multimodal speculative decoding. By integrating training-free sparse attention libraries and unified token pruning frameworks for MLLMs and audio tasks, AngelSlim delivers a production-ready infrastructure that maximizes inference throughput without compromising reasoning integrity. As an accessible and exhaustive toolkit, AngelSlim bridges the gap between algorithmic innovation and industrial-scale deployment, with a commitment to continuous functional expansion for the research community.

2 AngelSlim Quantization Framework

In this section, we will introduce the 2-bit quantization algorithm of the HY-1.8B-2Bit model and the ternary quantization, covering its data, training strategy, and quantization scheme. We will also introduce AngelSlim's PTQ framework, which supports a variety of 4–8 bit quantization algorithms and provides a one-click interface for developer convenience. Additionally, we will present LeptoQuant, an FP8 quantization algorithm designed for industrial deployment.

2.1 HY-1.8B-2Bit : An Efficient On-Device LLM

HY-1.8B-2Bit is a high-efficiency 2-bit LLM built for on-device deployment. It is derived from the Tencent Hunyuan-1.8B-Instruct model (Tencent HY Team, 2025b) via an optimized Quantization-Aware Training (QAT) framework (Liu et al., 2025), while preserving a lightweight dense architecture. To overcome the stringent resource constraints of edge hardware, HY-1.8B-2Bit utilizes Stretched Elastic Quantization (SEQ) (Liu et al., 2025), an aggressive ultra-low-bit strategy that compresses weights to a 2-bit representation. This results in a bit-equivalent model size of merely 0.3B, representing a $6\times$ compression compared to the original Hunyuan-1.8B model, which significantly alleviates inference latency and computational overhead. Despite the radical reduction in precision, HY-1.8B-2Bit retains the sophisticated Dual Chain-of-Thought (Dual-CoT) rea-

soning capability, achieving a significant average accuracy gain of 17% across multiple benchmarks compared to models with equivalent model size.

2.1.1 Model Strengths

Through the application of optimized quantization-aware training, HY-1.8B-2Bit achieves a performance profile that is remarkably competitive with the PTQ-INT4 variant, HY-1.8B-Instruct-Int4 (Tencent HY Team, 2025e). Comprehensive evaluations across diverse domains—including mathematics, humanities, and programming—reveal that HY-1.8B-2Bit incurs a marginal average performance degradation of only 4% relative to the full-precision Hunyuan-1.8B-Instruct (Tencent HY Team, 2025b) baseline. Given the $6\times$ compression ratio achieved, this minimal loss highlights the model's exceptional information retention and its capacity to maintain structural robustness even under radical bit-width reduction.

Superior Model Capability Through the application of QAT, HY-1.8B-2Bit achieves a performance profile that is remarkably competitive with the PTQ-INT4 variant, Hunyuan-1.8B-Instruct-INT4 (Tencent HY Team, 2025e). Comprehensive evaluations across diverse domains, including mathematics, humanities, and programming, reveal that HY-1.8B-2Bit incurs a marginal average performance degradation of only 4% relative to the full-precision Hunyuan-1.8B-Instruct (Tencent HY Team, 2025b) baseline. Given the $6\times$ compression, this minimal loss highlights the model's exceptional information retention and its capacity to maintain structural robustness even under radical bit-width reduction.

Efficiency Gains with Lightweight Scale HY-1.8B-2Bit redefines the efficiency frontier for small-scale language models. When benchmarked against dense models of equivalent bit-equivalent size, such as HY-0.5B-Instruct (Tencent HY Team, 2025a), HY-1.8B-2Bit maintains a substantial lead, outperforming its counterpart by an average of 16% across standard benchmarks. This performance gap suggests that knowledge from a larger model into an ultra-low-bit representation is more effective than training a smaller dense model from scratch. Consequently, HY-1.8B-2Bit serves as a scalable, high-efficiency alternative for edge computing, delivering sophisticated reasoning capabilities within a hardware-friendly footprint.

Comprehensive Reasoning Proficiency A distinguishing feature of HY-1.8B-2Bit is its inheritance of the "full-thinking" architecture from the HY-A13B (Tencent HY Team, 2025c), making it the most compact model in the industry to support complex reasoning pathways. By implementing a Dual Chain-of-Thought (Dual-CoT) strategy, HY-1.8B-2Bit allows for a dynamic trade-off between execution speed and cognitive depth. Users can invoke a concise "short-CoT" for straightforward inquiries to minimize latency, or a detailed "long-CoT" for tasks requiring rigorous logical multi-step synthesis. This dual-mode flexibility ensures that HY-1.8B-2Bit is uniquely suited for real-time, resource-constrained deployments where both high-fidelity logic and rapid response are non-negotiable requirements.

2.1.2 2-bit QAT Workflow

We present a high-efficiency QAT methodology specifically designed for HY-1.8B-2Bit. Our framework demonstrates the empirical feasibility of compressing high-precision LLMs into ultra-low bit-widths without incurring the "accuracy collapse" typically associated with extreme quantization. By leveraging low-bit QAT, we demonstrate that a model's capacity-to-size ratio can be elevated beyond the theoretical limits of traditional full-precision models of equivalent parameter counts, enabling the deployment of sophisticated intelligence in strictly resource-constrained environments.

Data Curative Pipeline. To ensure high-fidelity knowledge retention, we adapt the hierarchical data selection pipeline established in HY-A13B (Tencent HY Team, 2025c). This pipeline consists of three core stages: (1) Preprocessing: involving global deduplication, heuristic-based quality filtering, and denoising; (2) Model-Based Extraction: utilizing specialized classifiers to distill coherent text from raw corpora; and (3) Refinement: which applies semantic-level deduplication and rigorous quality assessments.

Model	CMMLU	C-Eval	ARC	BBH	GSM8K	HumanEval	LCB	GPQA	Average	Distance
HY-1.8B-FP16	55.07%	54.27%	70.50%	79.08%	84.08%	94.51%	31.50%	68.18%	67.15%	0.00%
HY-0.5B-FP16	37.08%	35.98%	49.89%	58.10%	55.04%	67.07%	12.11%	46.97%	45.28%	-21.87%
HY-1.8B-INT4	50.80%	48.67%	68.83%	74.80%	78.70%	89.02%	30.08%	65.56%	63.31%	-3.84%
HY-1.8B-2Bit	49.32%	47.60%	64.45%	75.54%	77.33%	93.29%	32.73%	65.15%	63.18%	-3.97%

Table 1: Benchmark performance comparison across diverse domains. All models are instruction-tuned. The "Distance" column denotes the accuracy gap relative to the 1.8B FP16 baseline.

We hypothesize that a compact, high-signal subset is superior to a massive, noisy corpus in the QAT process. Our preliminary experiments and previous works (Li et al., 2025b) indicate that performance decay in ultra-low-bit regimes is disproportionately concentrated in logical reasoning and long-context comprehension. To mitigate this, we curated a specialized hybrid dataset by increasing the proportion of scientific and mathematical tokens and integrating targeted long-form sequences. This resulted in **an optimized set of 89B tokens**, providing a high-density signal for recovering the model's cognitive capabilities during quantization.

Quantization Strategy Addressing the inherent instability of 2-bit precision, HY-1.8B-2Bit employs SEQ (Liu et al., 2025) to stabilize training. Traditional asymmetric INT2 mappings that include a zero value (e.g., $\{-2, -1, 0, 1\}$) often suffer from restricted dynamic range. In contrast, SEQ adopts a symmetric mapping scheme: $\{-1.5, -0.5, 0.5, 1.5\}$. By shifting the quantization centroid and eliminating the zero-point, SEQ optimizes the dynamic range coverage and resolves the "limited energy level" bottleneck. Coupled with an adaptive micro-tuning of the scaling factor for quantization intervals, this strategy significantly mitigates information dissipation, capturing high-dimensional features even within a 2-bit constraint.

QAT Configuration and Optimization In the 2-bit regime, QAT transitions from a simple "error compensation" task to a complex "distribution reconstruction" process, where weight manifolds undergo significant shifts to align with low-precision representations.

- **Initialization and Convergence:** Unlike frameworks such as BitNet (Ma et al., 2025), we initialize HY-1.8B-2Bit with instruction-tuned weights rather than raw pre-trained weights. This strategic initialization provides a superior starting point for the manifold reconstruction, accelerating convergence and drastically reducing the token budget required for recovery.
- **Hyperparameter "Wind Tunnel" Testing:** To navigate the sensitivity of 2-bit landscapes, we conducted extensive small-scale "wind tunnel" experiments using a 10B token subset. This allowed us to rapidly isolate optimal hyperparameters, bypassing the prohibitive computational cost of full-scale grid searches.
- **Data Efficiency:** We analyzed the performance returns of varying data volumes (30%, 50%, and 70% of the full SFT corpus). Our findings indicate that 50% of the SFT data provides an optimal equilibrium between accuracy recovery and computational overhead.

Notably, HY-1.8B-2Bit requires only 10% of the token consumption compared to models like BitNet-2B (Ma et al., 2025). This efficiency demonstrates that high-performance, ultra-low-bit models do not necessitate training from scratch, providing a scalable and cost-effective blueprint for the industrial-scale production of quantized LLMs.

2.1.3 Performance and Deployment Efficiency of HY-1.8B-2Bit

We evaluate HY-1.8B-2Bit across diverse standard benchmarks to assess its general knowledge (CMMLU (Li et al., 2024), C-Eval (Huang et al., 2023), GPQA Diamond Pass@3 (Rein et al., 2024)), logical reasoning (ARC (Clark et al., 2018), BBH (Suzgun et al., 2023)), and specialized technical proficiency (GSM8K (Cobbe et al., 2021), HumanEval Pass@3 (Chen et al., 2021), LiveCodeBench (Jain et al., 2024)). Performance is measured against the full-precision HY-1.8B model (Tencent HY Team, 2025b), an INT4 baseline (Tencent HY Team, 2025e) quantized via GPTQ (Frantar et al., 2022), and a scale-equivalent dense model, HY-0.5B (Tencent HY Team, 2025a).

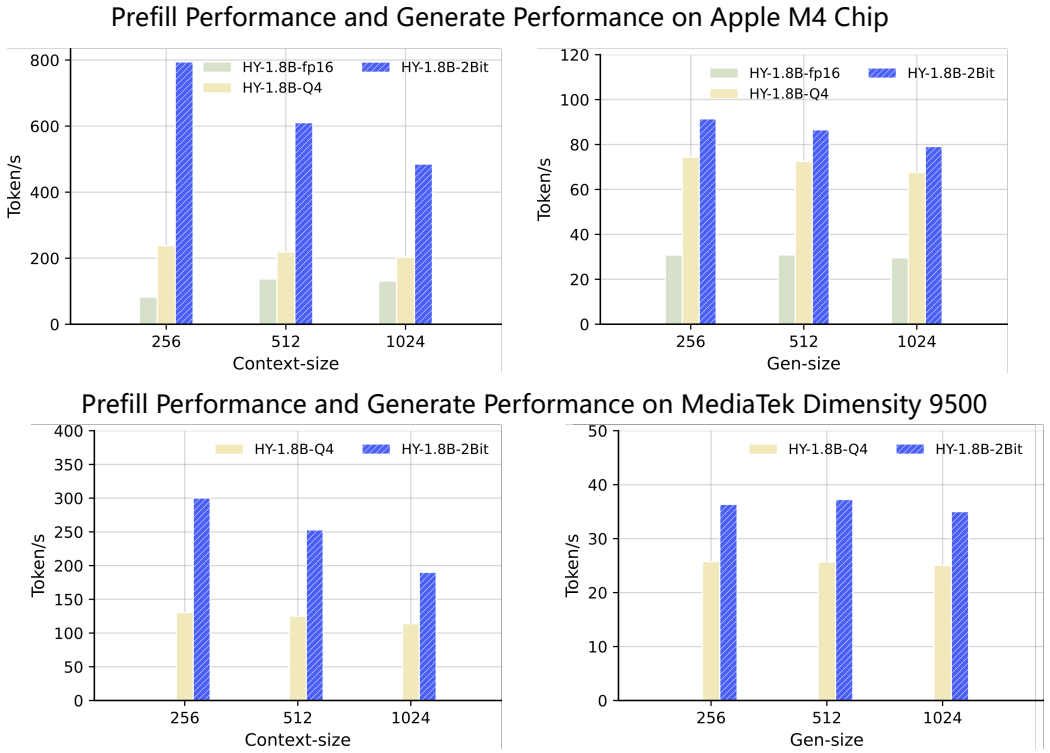

Figure 2: Comparison of inference latency (TTFT) and generation throughput on edge devices. HY-1.8B-2Bit consistently outperforms both 4-bit and FP16 baselines across varying sequence lengths.

Performance of HY-1.8B-2Bit: Table 1 summarizes the comparative performance of HY-1.8B-2Bit. The empirical results reveal that HY-1.8B-2Bit maintains high-tier performance despite the extreme reduction in bit-width, incurring a marginal average degradation of only 3.97% compared to its full-precision 1.8B teacher. Remarkably, HY-1.8B-2Bit performs nearly on par with the INT4 variant,with a negligible accuracy gap of only 0.13%, while utilizing only half the weight precision. In reasoning-intensive benchmarks such as BBH and LiveCodeBench, HY-1.8B-2Bit actually outperforms the 4-bit baseline. This suggests that our QAT framework effectively reconstructs the high-dimensional weight distributions necessary to preserve the complex logical structures of the base model, even under 2-bit constraints. When compared to the dense HY-0.5B model, which occupies a comparable model size, the superiority of the 2-bit QAT approach becomes evident. While the 0.5B dense model suffers a catastrophic 21.87% drop in average accuracy, HY-1.8B-2Bit remains robust, outperforming the smaller dense counterpart by 22.29% in GSM8K and 20.62% in LiveCodeBench. This significant gap validates our strategy of compressing a larger, high-capacity model into an ultra-low-bit format rather than training a smaller parameter model from scratch. This makes HY-1.8B-2Bit a competitive alternative for edge-based intelligence where memory and power budgets are strictly capped.

Efficiency of HY-1.8B-2Bit on Edge Hardware: We evaluate the practical deployment efficiency of HY-1.8B-2Bit on both Apple M4 and MediaTek Dimensity 9500 Chip. To standardize the benchmarking environment, we utilized a fixed configuration of 2 threads to measure Time-to-First-Token (TTFT) and generation throughput across varying context window sizes. For comparative analysis, we benchmarked the FP16, Q4_K_M (4-bit), and our HY-1.8B-2Bit GGUF formats. As illustrated in Figure 2, HY-1.8B-2Bit demonstrates substantial gains on the Apple M4 chip, achieving a $3\times$ to $8\times$ acceleration in TTFT for input sequences ranging from 256 to 1024 tokens compared to the BF16 baseline. Regarding generation speed, HY-1.8B-2Bit maintains a consistent $> 2\times$ speedup over BF16 across standard operating windows. On the MediaTek Dimensity 9500 Chip, the HY-1.8B-2Bit achieves a $\sim 2\times$ speedup compared to 4-bit model in prefill performance, while maintaining a $\sim 1.5\times$ in generative throughput. These results underscore HY-1.8B-2Bit's capacity for high-performance, real-time inference in resource-constrained environments.

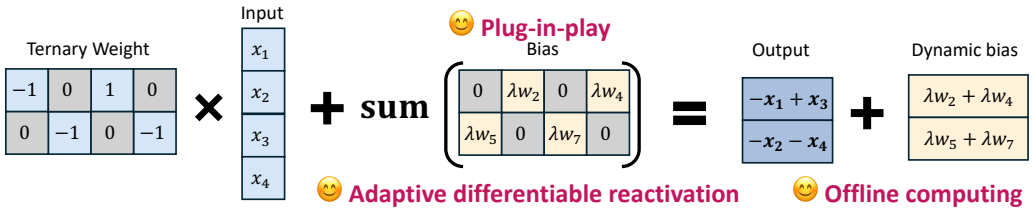

Figure 3: Tequila reactivates dead weights as adaptive dynamic biases via a differentiable function, achieving significant accuracy improvements with nearly zero inference overhead.

2.2 Ternary Quantization

The AngelSlim framework incorporates ternary quantization as a core strategy for extreme weight compression, constraining model weights to the discrete set $\{-1, 0, +1\}$. This paradigm effectively replaces computationally intensive floating-point multiplications with hardware-efficient additions via a lookup table-based engine like BitNet.cpp (Wei et al., 2025) and T-MAC (Wang et al., 2025). To overcome the traditional trade-offs between model accuracy and hardware alignment, AngelSlim introduces two specialized methodologies: **Tequila** (Huang et al., 2025) and **Sherry** (Huang et al., 2026).

2.2.1 1.58-Bit Tequila: Trapping-Free Ternary Quantization

Tequila addresses the *deadzone trapping* phenomenon inherent in standard ternary QAT. In conventional ternary quantization schemes (Ma et al., 2025), the quantization function for a weight vector $W = (w_1, \ldots, w_n)$ is typically defined as:

$$Q(W) = \hat{W}\alpha, \quad \hat{w}_i = \begin{cases} +1, & \text{if } w_i \geq \Delta; \\ 0, & \text{if } |w_i| < \Delta; \\ -1, & \text{if } w_i \leq -\Delta, \end{cases} \tag{1}$$

where $\hat{W}$ represents the ternary weights, α is a scaling factor, and Δ is the threshold. Due to the coarse nature of the Straight-Through Estimator (STE), weights within the "deadzone" $[-\Delta, \Delta]$ often receive uninformative gradients, leading to representational stagnation.

As illustrated in Figure 3, Tequila reactivates these trapped weights by repurposing them as biases during training:

$$Y = XQ(W) + C(W) = X\hat{W}\alpha + \sum_{i \in D} \lambda w_i, \tag{2}$$

where $D = \{i \mid |w_i| < \Delta\}$ denotes the set of indices in the deadzone. The bias term $C(W)$ acts as a residual connection, yielding an informative gradient signal for the dead weights:

$$\frac{\partial L}{\partial w_i} = x_i \frac{\partial L}{\partial Y} + \lambda \frac{\partial L}{\partial Y}, \quad \forall i \in D. \tag{3}$$

This formulation enables weights to escape the deadzone stably by providing a continuous signal in the forward pass and a direct gradient path in the backward pass. Crucially, these biases are merged into static network parameters offline post-training, ensuring nearly zero inference overhead while maintaining the theoretical ternary quantization.

2.2.2 1.25-Bit Sherry: Hardware-Efficient 3:4 Sparsification

Sherry resolves the architectural friction between non-standard ternary bit-widths and commodity hardware by introducing **3:4 fine-grained structured sparsity**. While standard ternary methods often force a choice between bit wastage (2-bit packing) or reduced inference throughput (1.67-bit irregular packing), Sherry restores power-of-two alignment,as shown in Figure 4.

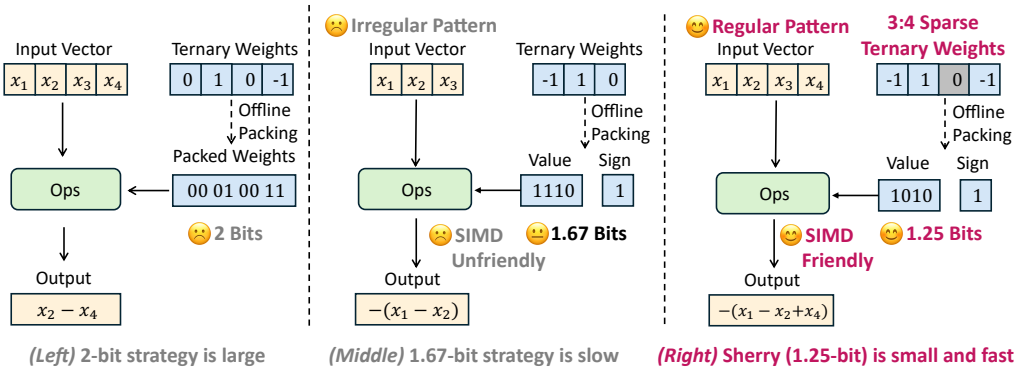

Figure 4: *(Left)* **2-bit strategy** packs each weight into 2 bits to maintain alignment, resulting in large bit wastage. *(Middle)* **1.67-bit strategy** packs 3 weights into 5 bits, introducing SIMD-unfriendly 3-way patterns, leading to slow speed. *(Right)* **Our Sherry** enforces a 3:4 sparsity and packs 4 weights into 5 bits, introducing SIMD-friendly 4-way patterns, achieving a small 1.25-bit width and faster inference speed.

3:4 Sparse Packing Sherry enforces a structural constraint where exactly three non-zero elements (± 1) are permitted within every contiguous block of four weights. This configuration allows for an optimal packing strategy where each block is stored in a compact 5-bit representation, as $N_{perm} = \binom{4}{3} \times 2^3 = 32$, which perfectly saturates a 5-bit index. This regularized mapping achieves a 1.25-bit width while maintaining native compatibility with Single Instruction Multiple Data (SIMD) vector lanes.

Arenas: Annealing Residual Synapse Directly imposing 3:4 sparsity on ternary weights frequently leads to *weight trapping*, where gradients become homogenized and weights accumulate in localized regions, causing representational collapse. To counteract this, Sherry introduces **Arenas**, a module that injects heterogeneous gradients during the training phase:

$$Y = XQ(W) + \lambda_t XW, \tag{4}$$

where λ_t is a scheduling coefficient that anneals to zero by the conclusion of training. By re-introducing the continuous latent weight W into the forward pass, Arenas prevents the gradients $\frac{\partial L}{\partial X}$ from collapsing into a low-rank state. This encourages diversity in weight updates and ensures the final sparse model retains high expressive capacity without any additional inference-time cost.

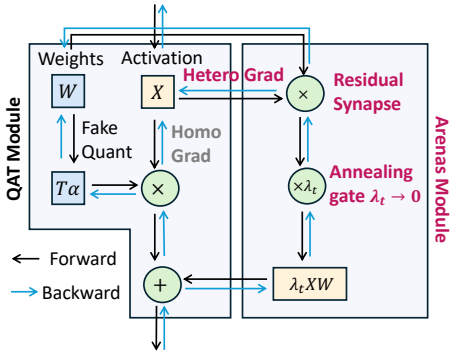

Figure 5: The overview of the Arenas module with QAT. The Arenas module injects the heterogeneous gradients through a residual synapse with an annealing gate.

2.2.3 Performance and Deployment Efficiency

We evaluate Tequila and Sherry on the LLaMA-3.2 1B and 3B models across multiple standard benchmarks to demonstrate their superior accuracy-efficiency trade-offs. The evaluation covers five zero-shot tasks: PIQA (Bisk et al., 2020), ARC-Easy (ARC-e), ARC-Challenge (ARC-c) (Clark et al., 2018), HellaSwag (HelS) (Zellers et al., 2019), and WinoGrande (WinG) (Sakaguchi et al., 2021). We compare our methods against several LLaMA-based ternary LLM baselines, including TWN (Li et al., 2016), Spectra (Kaushal et al., 2025), BitNet (Ma et al., 2024), TernaryLLM (Chen et al., 2024b), LLM-QAT (Liu et al., 2024b), and ParetoQ (Liu et al., 2025).

Performance of Tequila and Sherry The benchmark results in Table 2 demonstrate that both Tequila and Sherry significantly outperform existing ternary quantization baselines. For the 1B model series, while traditional ternary methods suffer from a catastrophic performance drop of over 16% on average, Tequila and

Model	Size	Bit-width	ARC-e	ARC-c	HelS	PIQA	WinG	Average
LLaMA3.2	1B	16	0.654	0.313	0.477	0.742	0.603	0.558
TernaryLLM*	1B	1.67	0.424	0.174	0.256	0.563	0.513	0.386
ParetoQ*	1B	1.67	0.421	0.180	0.273	0.604	0.510	0.398
LLM-QAT	1B	1.67	0.360	0.262	0.313	0.551	0.496	0.397
BitNet	1.3B	1.67	0.549	0.242	0.377	0.688	0.558	0.483
Spectra	1.1B	1.67	0.563	0.246	0.388	0.693	0.555	0.489
Tequila	1B	1.67	0.645	0.305	0.391	0.710	0.542	0.519
Sherry	1B	1.25	0.647	0.309	0.388	0.699	0.550	0.519
LLaMA3.2	3B	16	0.745	0.422	0.552	0.768	0.691	0.636
TernaryLLM*	3B	1.67	0.361	0.161	0.260	0.572	0.496	0.370
ParetoQ*	3B	1.67	0.498	0.231	0.303	0.645	0.529	0.441
LLM-QAT	3B	1.67	0.445	0.307	0.434	0.627	0.506	0.464
BitNet	3B	1.67	0.614	0.283	0.429	0.715	0.593	0.527
Spectra	3.9B	1.67	0.660	0.319	0.483	0.744	0.631	0.567
Tequila	3B	1.67	0.702	0.346	0.464	0.739	0.627	0.576
Sherry	3B	1.25	0.688	0.364	0.452	0.736	0.593	0.567

Table 2: Comparison of Tequila and Sherry with LLaMA-based ternary LLMs across different model sizes; * indicates results obtained from our reproduction.

Sherry reduce this gap to a marginal 3.9% relative to the BF16 LLaMA-3.2 1B baseline. In the 3B model series, Tequila trails the full-precision model by only 6.0%. Remarkably, Sherry achieves a 6.% gap despite utilizing a significantly lower bit-width of 1.25 bits. When compared to other high-performance ternary baselines like BitNet, SherryLLM achieves a 4% higher average accuracy while using 25% fewer bits. This demonstrates that our 3:4 structured sparsification and Arenas mechanism effectively mitigate representational collapse, allowing 1.25-bit models to match or exceed the cognitive capabilities of 1.58-bit counterparts.

Efficiency of Tequila and Sherry We evaluate the hardware efficiency on an Intel i7-14700HX CPU, comparing throughput (tokens per second) and model size as shown in Table 3. Sherry consistently exhibits the highest efficiency across all scales. By restoring power-of-two alignment through its 5-bit packing strategy, Sherry achieves a generation speed of 148.27 t/s for the 0.7B scale, surpassing BitNet (I_S, 2.0 bits) and Tequila (TL2, 1.67 bits). Furthermore, Sherry reduces the model size by approximately 20% compared to 2-bit counterparts. These results confirm that AngelSlim's hardware-aligned sparsification provides a superior "sweet spot" for deploying high-performance LLMs on edge devices with limited memory bandwidth.

Scale	Method	Bits	Speed (t/s)	Size (MB)
0.7B	BF16	16	34.01	1360.0
	BitNet(I2_S)	2.0	132.13	256.56
	Tequila(TL2)	1.67	116.83	233.44
	Sherry	1.25	148.27	205.50
3B	BF16	16	7.55	6190.0
	BitNet(I2_S)	2.0	41.87	873.65
	Tequila(TL2)	1.67	38.80	846.01
	Sherry	1.25	45.55	712.40

Table 3: Inference efficiency comparison on Intel i7-14700HX.

2.3 Post-training Quantization

Post-training Quantization (PTQ) serves as a critical enabling technology for deploying large-scale neural networks efficiently, as it transforms high-precision models into compact, inference-optimized forms without requiring retraining. In this section, we will first introduce the PTQ framework of AngelSlim, and then present our proposed LeptoQuant.

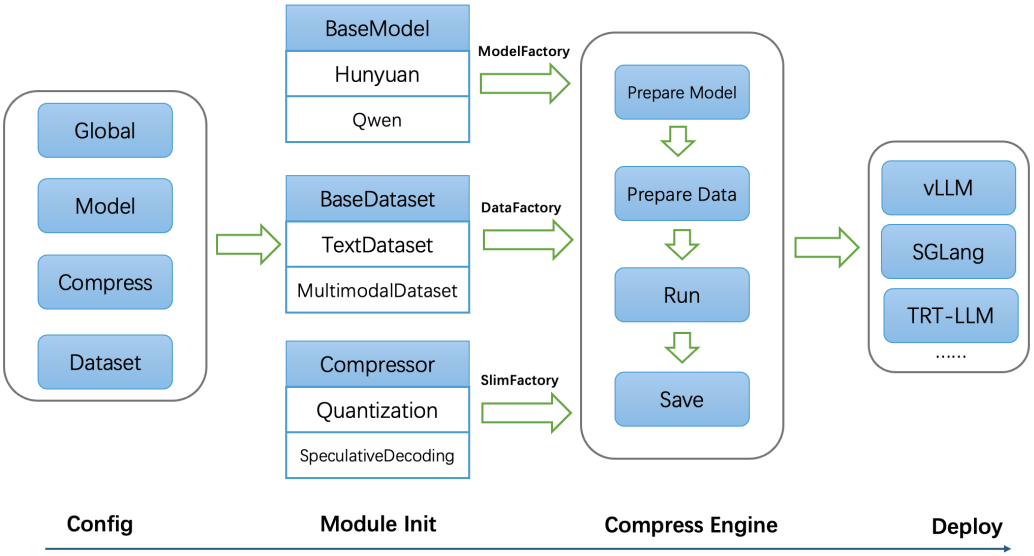

Figure 6: AngelSlim Quantization Framework.

2.3.1 PTQ Framework

To address the storage and computational bottlenecks in deploying large models, Angelslim provides a unified post-training quantization (PTQ) framework. The framework not only integrates current mainstream quantization schemes but also offers a complete toolchain for analysis, ensuring the reliability of quantized models. Our AngelSlim framework is characterized by the following key features:

- Highly Integrated: This toolkit integrates mainstream compression algorithms into a unified framework, offering developers one-click access with exceptional ease of use.
- Continuous Innovation: Beyond integrating widely-used industry algorithms, we are continuously researching better compression algorithms, which will be gradually open-sourced in the future.
- Performance-Driven: We continuously optimize end-to-end performance in model compression workflows and algorithm deployment, such as enabling quantization of models like Qwen3-235B and DeepSeek-R1 on a single GPU.

As shown in Figure 6, AngelSlim starts by parsing a YAML configuration file to load all essential parameters for the compression task. This includes global settings, model information, compression algorithm specifications, and dataset configurations. This centralized configuration approach enables users to customize the pipeline without modifying the core code, ensuring high flexibility and reproducibility.

Moving to the Module Init stage, the framework initializes three specialized factories to encapsulate its core components: the ModelFactory manages the registration and instantiation of base models such as Hunyuan (Tencent HY Team, 2025c) and Qwen (Yang et al., 2025a), allowing seamless integration of new LLMs through a registration mechanism; the DataFactory, built on a DataLoaderFactory, encapsulates diverse dataset types including TextDataset and MultimodalDataset to create unified data loaders, ensuring consistent data preprocessing across tasks; and the SlimFactory provides access to a suite of compression techniques such as quantization and speculative decoding, which are registered and dispatched dynamically based on the configuration to support multiple compression strategies in a unified manner.

The Compress Engine stage then orchestrates the end-to-end compression workflow: it first prepares the target model from the ModelFactory and initializes the data loader from the DataFactory, then executes the selected compression algorithm from the SlimFactory to reduce model size and latency while preserving performance, before saving the compressed model checkpoint for deployment. Finally, the compressed models can be directly deployed via high-performance inference backends, including vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024). Together, these stages form a streamlined, extensible pipeline that unifies model compression and deployment, making it adaptable to a wide range of LLM optimization scenarios.

Model	Quantization	GPQA Diamond	AIME 2024	SimpleQA	LiveCodeBench
DeepSeek-R1-0528	FP8-Block-Wise	78.28	88.67	27.8	77.1
	W4A8-FP8	77.37	88.67	26.83	78.86

Table 4: Benchmark results for DeepSeek-R1-0528 model with FP8-Block-Wise and W4A8-FP8 quantization algorithms on datasets including GPQA Diamond, AIME 2024, SimpleQA and LiveCodeBench

At the algorithmic level, the framework covers the full precision spectrum from FP8, INT8 to INT4. FP8 quantization focuses on achieving significant speed-up with limited precision loss, supporting both dynamic and static modes: Dynamic Quantization (W8A8-FP8 Dynamic) performs real-time scaling on activation values to adapt to input variations; whereas Static Quantization (W8A8-FP8 Static) determines a unified scaling factor via offline calibration, pursuing ultimate inference speed. For extremely large models, its unique Low-Memory FP8 Calibration mode effectively reduces the GPU memory overhead of the quantization process itself. For higher compression demands, the framework provides two mainstream INT4 weight quantization schemes: The AWQ (Lin et al., 2024) algorithm adaptively amplifies the numerical range of important weight channels by analyzing activation distributions, thereby preserving more information under 4-bit quantization; the GPTQ algorithm employs layer-wise reconstruction optimization to minimize the output error after quantization. Both require only a small amount of calibration data, operate without training, and achieve a balance between accuracy and compression efficiency.

To ensure quantization quality, the framework includes built-in diagnostic analysis tools. Users can enable Scale Analysis during quantization to detect outliers, and after quantization, use dedicated tools to compare the distribution differences between FP8 and the original BF16 weights, providing data insights for potential accuracy restoration.

The framework's practicality is reflected in its broad adaptation to mainstream model families. For the Hunyuan series models, it primarily supports FP8, INT4-AWQ, and INT4-GPTQ schemes. For the DeepSeek-R1 (Liu et al., 2024a) model, a more aggressive mixed-precision strategy is further implemented: its W4A8-FP8 scheme performs group-wise INT4 quantization on weights (group size 128) while keeping activation values in FP8 precision, achieving in-depth optimization of storage and computational efficiency; its INT4-AWQ scheme can also seamlessly integrate with inference engines like vLLM. Additionally, the framework is compatible with diverse Qwen series models, including multimodal models, demonstrating good generalization capability.

In addition, the Angelslim framework takes into account the hardware resource limitations in practical deployment. Its Low-Memory calibration mode implements an intelligent CPU-offloading strategy: instead of keeping all intermediate layer parameters in GPU memory, it strategically stores them in the much larger CPU RAM, loading them back to the GPU only when computation is required. This on-demand swapping mechanism significantly reduces the peak GPU memory overhead of the quantization process itself while ensuring calibration accuracy. This enables large-scale models like DeepSeek-R1 to complete the entire calibration process using only a single GPU, without relying on multi-GPU or high-memory configurations.

Beyond memory efficiency, the framework's core strength lies in its ability to preserve model accuracy under aggressive quantization. Table 4 demonstrates its efficacy, showcasing near-lossless accuracy retention for DeepSeek-R1 even under highly compressed W4A8 schemes, which is a significant challenge given the model's scale and reasoning capabilities.

In summary, by constructing a complete ecosystem encompassing multiple algorithms, granularities, and models, and equipped with analysis tools, the Angelslim PTQ framework provides a practical, engineering-ready solution for the efficient deployment of large-scale language models.

2.3.2 LeptoQuant

For PTQ (Post-Training Quantization), In this section, we analyzed the numerical precision loss experienced by models after PTQ quantization to FP8. Then we introduced the Dynamic Outlier Isolation Scale (LeptoQuant), which addresses this anomalous loss by searching for optimal scaling values to mitigate quantization loss. The figure shows the overall architecture.

Typically, PTQ calculates the abs Max values of activations and weights as quantization scaling factors. By observing the numerical distribution of models after FP8 PTQ quantization, we found that the variance of activation distributions is significantly greater than that of bf16 models. This numerical distribution causes

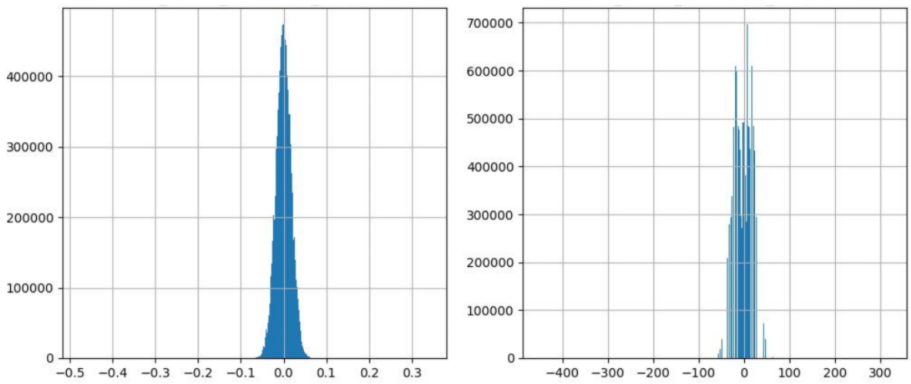

Figure 7: Figure a is the BF16 weight distribution histogram of hunyuan-4b-insturct, and Figure b is the weight distribution histogram after FP8 quantization. It can be seen that the weight distribution after quantization is slightly smoothed to a position far away from 0.

quantized values to fall within a range that is difficult to represent in FP8, resulting in excessive losses on mathematically difficult tasks or those requiring strict text formatting.

Figure 7 shows a histogram of the numerical distribution of the last layer of $FFN2$ before and after quantization in a model with significant FP8 quantization loss. By observing the original precision numerical distribution, it is found that the overall numerical distribution of the weight is concentrated in a peak distribution, with obvious outliers and most of the data concentrated near zero. The relative distance between the values is small and the data distribution approaches the laplace distribution, resulting in higher precision requirements during the calculation process. The weight distribution after FP8 QDQ is shown in the figure on the right. It can be seen that the quantized distribution is smoother than the original precision. Since the numerical expression of FP8-E4M3 is closer to 0, the more values can be represented, and the original precision weight under the normal distribution is close to Worigin. Traditional FP8 quantization causes the originally densely populated numbers to be smoothed to the FP8 precision range with poor expressiveness, resulting in reduced precision expression and loss of performance.

To address the aforementioned FP8 quantization issues, we introduced Leptokurtic Quant (LeptoQuant) search, a search strategy that concentrates the FP8 weight mapping range into a high-precision region by isolating outliers. Quantizing activations is generally more difficult than quantizing weights, so we prioritize optimizing FP8 activations. LeptoQuant uses the original FP8 outlier values as the upper limit of FP8 precision expression, calculating a new scale that compresses the value distribution into a high-precision distribution range, resulting in better precision expression of quantized activation values.

Searching to scale. Specifically, we define the FP8 value obtained by traditional PTQ calculation as $FP8_{origin}$, and then introduce a hyperparameter α, which determines the proportion of isolated outliers. Generally speaking, the quantization of activation values is much more difficult than weights, so we will prioritize optimize the activation. Our search range is from M to 0. For a given α, we calculate the outlier value of $FP8_{origin}$ as the new scaling factor denominator D, thereby constraining the FP8 quantization expression range and compressing the values in the densely distributed area to a position with higher precision. The expression of D is as follows:

$$D_w = Outlier(W, \alpha), \quad W = \max|FP8_{origin}|, \tag{5}$$

where α is selected to start searching from the 0.1% outlier, $Outlier$ represents the value of position a in W given the weight and α.

Calculating the quantization error. We calculate the loss of each block layer by layer through dynamic interpolation. Consider a FFN block, we obtain the activation and weight of the model, and the calculation of the FFN block can be written as $O = block(x)$. Then we introduce $formula$ into the block forward reasoning process to simulate the actual FP8 quantization calculation results of $LeptoQuant$. Specifically, the formula for quantifying $FFN1$ can be defined as:

$$\hat{Y}_D = Block(w, x, Scale_D), \quad Scale_D = \frac{w_{max}}{D_w}, \tag{6}$$

Base Model	Type	OlympiadBench	AIME_2024	AIME_2025
Hunyuan-4B-Instruct	BF16	73.10	78.30	66.50
	FP8	72.40	66.70	46.70
	FP8-lepto	73.10	76.66	60.70
Hunyuan-2B-Instruct	BF16	73.41	56.67	53.85
	FP8	72.41	50.00	37.00
	FP8-lepto	72.64	53.00	36.00

Table 5: Compare the performance of the standard FP8 with that of the HY series models using leptoQuant

Base Model	Type	GSM8K-flexible-extract	GSM8K-strict-match	HUMANEVAL
Qwen2.5-1.5B-Instruct	BF16	59.00	54.20	37.20
	FP8	55.72	49.36	32.93
	FP8-lepto	57.09	52.31	35.37
QWEN3-0.6B	BF16	41.62	41.7	20.12
	FP8	37.83	38.06	20.73
	FP8-lepto	38.59	38.44	21.95
QWEN3-4B	BF16	84.91	85.44	73.17
	FP8	83.85	83.85	67.7
	FP8-lepto	83.62	83.93	68.9

Table 6: Compare the performance of the standard FP8 with that of the Qwen series models using leptoQuant

where D_w is the denominator of the scaling factor of activation w, $Scale_D$ is the FP8 quantization scaling factor, and $\hat{Y}_D$ is the block-wise calculation result after quantizing the activation to FP8. Therefore, for each OP, we choose to automatically search for an optimal scaling factor to minimize the output error after quantization of a specific block. Formally, we want to optimize the following objective:

$$L(s) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \quad \{i \mid i \in \mathbb{Z}, 0 \leq i \leq n_{samlps}\}, \tag{7}$$

where i indicates different samples. For the search process, we use a more stable search space, fast grid search to select α in the interval $[0, 0.001]$ (0 means traditional FP8 quantization without isolating outliers, and 0.001 means the most aggressive scaling in the search space) and calculate the optimal α using the MSEloss.

LeptoQuant Benchmark For open-source models, we conducted LeptoQuant experiments on models and datasets from the Hunyuan and Qwen series that showed significant performance degradation after FP8 quantization, as shown in the Table. 6 and 5 .

3 Speculative Decoding

3.1 Training Framework

Speculative decoding has emerged as a critical technique for accelerating LLM inference; however, the training of draft models remains poorly supported by existing frameworks. Unlike standard language modeling, draft model training is inherently *target-model–dependent*: the objective is not to optimize standalone generation quality, but to maximize alignment with the token distribution and generation dynamics of a fixed target model. This dependency introduces requirements—such as target-model feedback, multi-step prediction supervision, and customized execution flows—that are difficult to express within conventional training pipelines.

These challenges are further amplified by advanced methods such as Eagle-3. Its tree-based attention structure and multi-step verification impose strict constraints on data ordering, attention masking, and temporal consistency during training. Meanwhile, the rapid evolution of large-scale architectures—including Mixture-of-Experts (MoE) and long-context Transformers—demands training solutions that are both scalable and compatible with production inference engines. Existing approaches typically address these aspects in isolation, resulting in fragmented systems that are difficult to maintain, extend, or deploy.

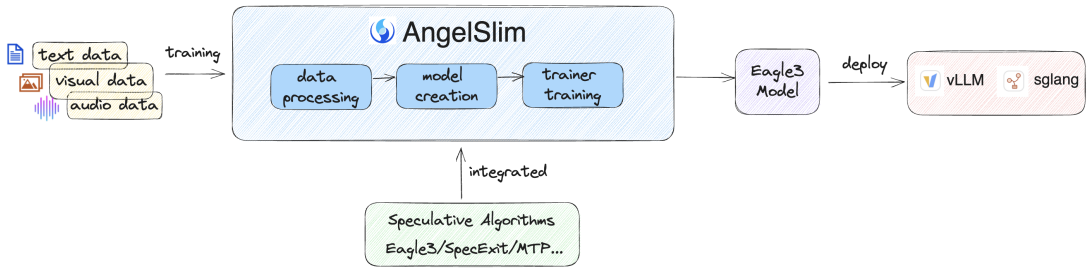

Figure 8: AngelSlim speculative training framework. The framework unifies data processing, model construction, and trainer-level execution to support *target-model–dependent* draft model training.

To address these limitations, we design a dedicated speculative sampling training framework in AngelSlim, with Eagle-3 compatibility, production inference integration, and architectural scalability as first-class design goals.

3.1.1 Overview and Objective

AngelSlim is a unified model compression toolkit that integrates multiple acceleration techniques—including quantization and speculative decoding—across language, vision–language, and speech models. In this section, we focus on its speculative sampling training framework, which treats speculative decoding as a *first-class training objective* rather than a post-hoc inference optimization.

Speculative sampling accelerates inference by decoupling generation and verification: a lightweight draft model predicts multiple future tokens in parallel, while a target model verifies these predictions in a single step. AngelSlim operationalizes this paradigm through an end-to-end training pipeline—from data preparation and model abstraction to algorithm-specific execution—explicitly optimizing draft models to align with the target model's token distribution and generation dynamics.

By design, the framework supports Eagle-3–style training while remaining deployment-oriented and modality-agnostic, enabling draft models trained in AngelSlim to be directly integrated into production inference engines.

3.1.2 Design Principles

AngelSlim's speculative sampling framework is built around three core principles:

Eagle-3 train-and-deploy compatibility. Draft models trained in AngelSlim can be directly deployed through the speculative decoding interfaces of production inference engines such as vLLM and SGLang, without additional conversion or parameter tuning.

Multi-modality by design. Text, vision–language, and speech models share a unified set of training abstractions and algorithmic components, avoiding modality-specific reimplementation.

Scalability and extensibility. The framework natively supports large-scale architectures, including MoE models and long-context Transformers, enabling efficient training across a wide range of model sizes.

Across diverse models and modalities, AngelSlim-trained Eagle-3 draft models achieve a consistent **1.4×–1.9× inference speedup** under standard speculative decoding settings.

3.1.3 Core Training Components

Data Processing Module The data pipeline provides a reusable and stable foundation for speculative training across modalities. It includes:

- Data resampling to construct in-distribution training data aligned with the target model.
- Unified preprocessing that converts text, image, and audio inputs into standardized token IDs and loss masks, including vocabulary mapping for pruned draft models.

Model	Method	GSM8K		Alpaca		HumanEval		MT-bench		Mean	
		TPS	AL	TPS	AL	TPS	AL	TPS	AL	TPS	AL
Qwen3-1.7B	Vanilla	376.42	1	378.86	1	378.38	1	390.53	1	381.05	1
	Eagle3	616.9	2.13	653.29	2.19	680.1	2.2	621.44	2.17	642.93	2.17
Qwen3-4B	Vanilla	229.05	1	235.29	1	234.66	1	234.04	1	233.26	1
	Eagle3	389.35	2.07	395.97	2.1	377.84	2.08	384.6	2.07	386.94	2.08
Qwen3-8B	Vanilla	149.63	1	149.93	1	153.85	1	153.81	1	151.81	1
	Eagle3	257.32	2	266.69	2.02	244.89	1.97	258.2	1.97	257.52	1.99
Qwen3-14B	Vanilla	92.97	1	92.66	1	92.94	1	94.46	1	93.26	1
	Eagle3	153.72	1.87	140.46	1.78	144.68	1.76	142.45	1.74	145.33	1.79
Qwen3-32B	Vanilla	43.49	1	43.38	1	43.19	1	43.3	1	43.32	1
	Eagle3	80.43	2.01	72.49	1.9	71.57	1.86	74.1	1.86	74.1	1.91
Qwen3-30B-A3B	Vanilla	311.84	1	320.43	1	325.77	1	325.42	1	320.87	1
	Eagle3	453.97	2.1	432.45	2.04	428.81	2.02	437.06	2.01	438.07	2.04

Table 7: Qwen3 series models benchmark using Eagle3 speculative decoding on vLLM (tp=1, ep=1, num_speculative_tokens=2, batch_size=1, output_len=1024). *TPS: throughput (tokens/s); AL: average accepted speculative tokens per decoding step.*

Model	Method	MT-Bench		MATH-500		MMMU		MMStar	
		TPS	AL	TPS	AL	TPS	AL	TPS	AL
Qwen3-VL-2B-Instruct	Vanilla	346.31	1	82.96	1	83.27	1	81.63	1
	Eagle3	555.22	2.29	163.09	2.57	154.18	2.55	139.73	2.31
Qwen3-VL-4B-Instruct	Vanilla	212.10	1	67.96	1	65.88	1	67.75	1
	Eagle3	382.33	2.34	141.87	2.72	104.44	2.05	107.07	2.10
Qwen3-VL-30B-A3B-Instruct	Vanilla	180.57	1	31.08	1	31.51	1	30.93	1
	Eagle3	240.47	2.19	75.31	2.79	48.47	1.78	52.57	1.94

Table 8: Qwen3-VL series models benchmark using Eagle3 speculative decoding on vLLM (tp=1, ep=1, num_speculative_tokens=4, batch_size=1, output_len=1024). *TPS: throughput (tokens/s); AL: average accepted speculative tokens per decoding step.*

- Hidden state extraction from the target model, serving as supervision signals for draft model training.

Model Abstraction Module Model extensibility is enabled through a unified `TargetModel` interface, which abstracts:

- Model loading and weight management
- Forward execution
- Intermediate and hidden state access

To support a new architecture or backend, users only need to implement the required abstract methods, without modifying the trainer or algorithm logic. This design significantly reduces the engineering overhead required to adapt AngelSlim to new models and modalities.

Trainer Module The trainer encapsulates the algorithm-specific logic of Eagle-3 and supports two execution modes:

- **Online training**, where hidden states are computed on the fly, suitable for smaller models or memory-rich environments.
- **Offline training**, where hidden states are precomputed and stored, enabling efficient training of large models under limited GPU memory.

To ensure stability and correctness during training, the trainer further provides:

Model	Dataset	Method	TPS	AL
Hunyuan-OCR	OmniDocBench	Vanilla	70.12	1
		Eagle3	108.1	2.08
Qwen2-Audio	LibriSpeech	Vanilla	78.76	1
		Eagle3	146.66	3.51
Fun-CosyVoice3	LibriTTS	Vanilla	-	1
		Eagle3	-	1.96

Table 9: Benchmark results of Eagle3 speculative decoding on vLLM (Hunyuan-OCR & Qwen2-Audio & Fun-CosyVoce, tp=1, ep=1, num_speculative_tokens=4, batch_size=1, output_len=1024). *TPS: throughput (tokens/s); AL: average accepted speculative tokens per decoding step.*

- **Training-time testing**, which explicitly simulates multi-step speculative generation so that the draft model learns to condition on its own predictions.
- **Native checkpointing support**, enabling full recovery of model parameters, optimizer states, and training progress.

These safeguards ensure that the resulting draft models behave robustly under speculative decoding at inference time. In practice, AngelSlim-trained models can be directly deployed in vLLM-based inference pipelines, consistently improving acceptance length and end-to-end throughput across diverse tasks.

3.1.4 End-to-End Performance

We evaluate end-to-end performance across a diverse set of benchmarks, covering language-only, vision–language, OCR, and speech tasks. All experiments employ Eagle-3 speculative decoding on vLLM with tp=1, ep=1, and task-specific numbers of speculative tokens.

For the Qwen3 series (Table 7), Eagle-3 consistently improves throughput (TPS) across all model sizes relative to vanilla decoding. The average accepted speculative length (AL) ranges from 1.74 to 2.2, indicating effective multi-token speculation while preserving correctness. Larger models exhibit particularly strong speedups, suggesting favorable scaling behavior.

In multimodal settings, the Qwen3-VL series (Table 8) similarly benefits from Eagle-3 across MT-Bench, MATH-500, MMMU, and MMStar. Smaller models achieve AL values above 2.3, while the largest model shows slightly reduced AL on certain benchmarks, reflecting the trade-off between speculative depth and verification cost at scale.

Non-language tasks also demonstrate substantial gains. Hunyuan-OCR on OmniDocBench (Table 9) improves throughput from 70.12 to 108.1 TPS with an AL of 2.08. Speech models, including Qwen2-Audio on LibriSpeech and Fun-CosyVoice3 on LibriTTS, similarly achieve significant throughput improvements while maintaining high acceptance rates.

Overall, Eagle-3 speculative decoding delivers a consistent **1.8×–2.0× end-to-end speedup** across all AngelSlim-trained models, demonstrating its effectiveness as a scalable and reliable acceleration technique for both language and multimodal inference.

3.2 Algorithmic Innovations

In addition to the system-level and framework advancements presented earlier, we also draw inspiration from recent algorithmic innovations in the efficient acceleration of large reasoning models. One notable example is *SpecExit* (Yang et al., 2025b), a novel approach designed to address inefficiencies caused by "overthinking" in large reasoning models (LRMs). LRMs often generate excessively long reasoning chains, which, while improving accuracy, result in high latency and unnecessary computation. Traditional early-exit mechanisms mitigate this issue by terminating generation once a sufficient answer is formed, but they typically introduce detection overhead and lack generalizability.

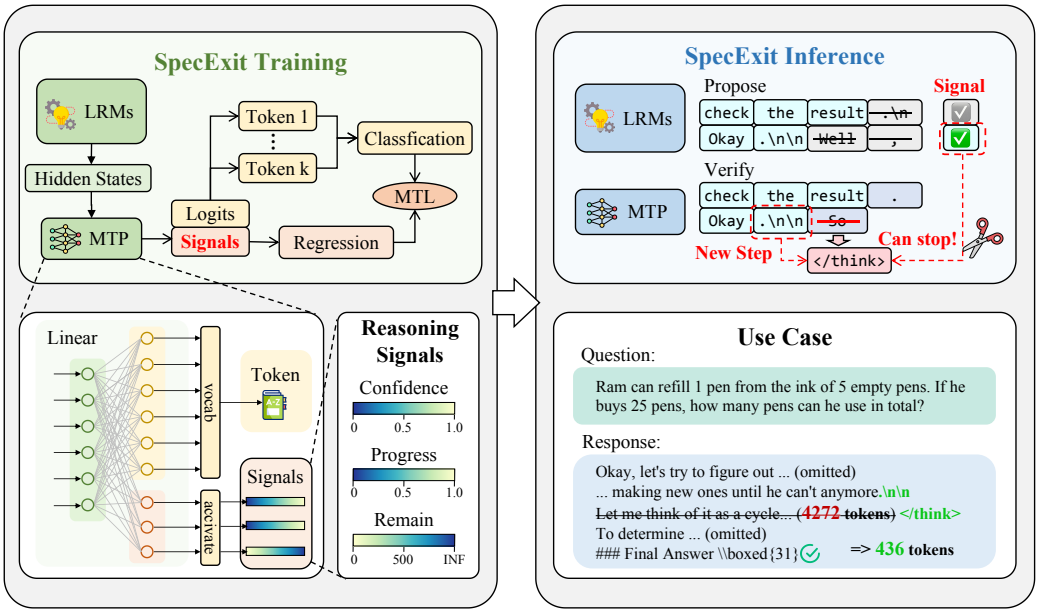

Figure 9: Overall architecture of the proposed SpecExit framework. The Multi-Token Prediction (MTP) layer is augmented to output both token logits and auxiliary signals. Training is performed with multi-task learning, while at inference these signals guide speculative early stopping without modifying the backbone model. The example illustrates how redundant reasoning steps can be pruned while preserving final answer quality.

SpecExit overcomes these limitations by integrating early-exit prediction directly into a lightweight draft model's hidden states. This enables the model to simultaneously propose future tokens and generate early-exit signals without additional probing overhead. The method leverages latent signals such as confidence, reasoning progress, and remaining steps to dynamically determine when to exit the reasoning process early. As a result, SpecExit can significantly reduce the average generation length—reported to decrease by approximately 66%—and achieve substantial end-to-end latency improvements, with up to a 2.5× speedup compared to speculative decoding baselines, while maintaining task accuracy.

By exploiting hidden-state information for dynamic reasoning control, SpecExit exemplifies how algorithmic innovations can be combined with speculative decoding techniques to accelerate large model inference in practical applications.

3.2.1 Method: SpecExit

SpecExit extends speculative decoding by embedding early-exit decision making directly into the draft model, enabling dynamic regulation of reasoning depth without introducing additional detection modules or modifying the backbone language model. The core insight is that the hidden states used for next-token prediction also encode higher-level information about reasoning progress, confidence, and content completeness. To make these signals explicit and learnable, SpecExit augments the draft model's Multi-Token Prediction (MTP) layer to jointly predict speculative token logits and auxiliary reasoning-related signals. In addition to projecting hidden states into the vocabulary space for multi-token prediction, the extended MTP layer includes lightweight auxiliary heads that estimate confidence, reasoning progress, and remaining reasoning length, while preserving the draft model's original language modeling capability.

Figure 9 illustrates the overall SpecExit architecture and workflow. During training, the extended MTP layer is optimized via multi-task learning, jointly supervising token prediction and auxiliary signal estimation. At inference time, the draft model simultaneously proposes speculative tokens and corresponding exit signals in a single forward pass. These signals guide early stopping decisions within the speculative decoding loop: if the predicted reasoning state indicates sufficient completeness, generation terminates early; otherwise, speculative tokens are verified by the target model as in standard speculative decoding. By integrating early-exit prediction directly into the draft–verify pipeline, SpecExit retains the low-latency benefits of speculative decoding while

Method	Math									Coding			Science			Logic		
	GSM8K			MATH500			AIME			HUMANEVAL+			GPQA-D			ARC-Challenge		
	Acc↑	Tok↓	Lat↓	Acc↑	Tok↓	Lat↓	Acc↑	Tok↓	Lat↓	Acc↑	Tok↓	Lat↓	Acc↑	Tok↓	Lat↓	Acc↑	Tok↓	Lat↓
Qwen3-4B-Thinking-2507																		
Think	95.3	1414	155.6	96.6	6719	530.1	86.7	19577	243.3	90.9	5079	175.3	68.7	9041	325.8	95.6	1812	156.5
*NoThink**	95.2	1631	204.2	96.6	6395	488.5	86.7	19816	243.2	88.4	4480	131.5	67.2	8833	276.8	95.1	1889	159.8
DEER	94.3	960	230.3	94.4	4893	519.6	70.0	17838	218.6	86.6	4079	242.4	67.2	9053	505.2	94.6	1011	200.3
EAGLE3	94.8	1408	140.3	96.6	6670	395.7	80.0	19792	187.3	87.2	5178	81.7	67.7	8975	212.2	95.7	1822	164.2
SpecExit	**93.8**	**649**	**75.8**	**96.8**	**4777**	**367.9**	**90.0**	**17769**	**206.1**	**89.6**	**4319**	**58.4**	**68.7**	**7011**	**137.0**	**94.5**	**588**	**71.4**
DeepSeek-R1-Distill-Llama-8B																		
Vanilla	76.4	1008	629.4	81.8	6878	857.1	36.7	22170	307.0	74.4	6287	445.5	43.6	8857	574.0	49.9	1917	628.5
NoThink	54.6	233	22.2	55.2	1643	262.8	10.0	8744	184.1	46.3	472	7.3	26.8	1200	166.6	12.6	135	13.6
DEER	74.7	710	484.8	80.8	3533	973.3	40.0	15619	272.3	79.3	4206	269.2	40.9	8492	521.5	47.5	1029	531.3
EAGLE3	79.3	976	276.9	80.8	6172	593.6	30.0	25686	228.1	78.7	5312	346.5	43.9	8749	420.1	59.2	1378	496.4
SpecExit	**75.3**	**333**	**112.6**	**80.6**	**1968**	**348.3**	**36.7**	**8160**	**176.0**	**81.7**	**3105**	**118.1**	**46.0**	**6849**	**307.5**	**50.3**	**500**	**253.7**

Table 10: Performance comparison of various reasoning methods on mathematical, scientific, general, and coding benchmarks. "Acc" denotes accuracy, "Tok" denotes token count, and "Lat" denotes total end-to-end latency. ↑ indicates higher is better; ↓ indicates lower is better.

avoiding the overhead and brittleness of external stopping criteria, and can be deployed without any changes to the target model.

3.2.2 Experiments

We evaluate SpecExit on a diverse set of reasoning benchmarks spanning mathematics (GSM8K, MATH500, AIME), coding (HumanEval+), science (GPQA-D), and logical reasoning (ARC-Challenge). Experiments are conducted on two representative large reasoning models, Qwen3-4B-Thinking-2507 and DeepSeek-R1-Distill-Llama-8B, and compared against standard reasoning and inference-time acceleration baselines, including speculative decoding with EAGLE3. Table 10 reports accuracy, generated token count, and end-to-end latency.

Across all benchmarks and models, SpecExit consistently achieves substantial reductions in reasoning length and inference latency while maintaining comparable accuracy. For Qwen3-4B-Thinking-2507, SpecExit reduces generated tokens by up to 54% on GSM8K and 53% on ARC-Challenge, translating into nearly 2× end-to-end latency reduction compared to the EAGLE3 baseline. On DeepSeek-R1-Distill-Llama-8B, the effect is even more pronounced: SpecExit shortens reasoning trajectories by up to 66% and delivers up to 2.5× latency speedup on GSM8K, despite aggressive pruning of intermediate reasoning steps.

Importantly, these efficiency gains incur only marginal accuracy differences, confirming that SpecExit primarily removes redundant reasoning rather than truncating essential computation. In contrast to heuristic or rule-based early stopping methods, SpecExit achieves a more favorable trade-off between reasoning depth and latency by leveraging learned exit signals integrated into the speculative decoding pipeline. Overall, the results demonstrate that SpecExit is an effective and practical mechanism for accelerating large reasoning model inference without sacrificing task performance.

4 Sparse Attention and Token Pruning

In this section, we present our approach to optimize the computational efficiency of LLMs and MLLMs through structural and dynamic sparsity. As sequence lengths continue to scale, the self-attention mechanism becomes a primary bottleneck due to its $O(n^2)$ complexity. To address this challenge, AngelSlim proposes a two-pronged strategy:

Sparse Attention for Long-Sequence Prefill: A framework specifically optimized to accelerate the KV cache generation stage in LLMs by exploiting the intrinsic sparsity of attention maps.

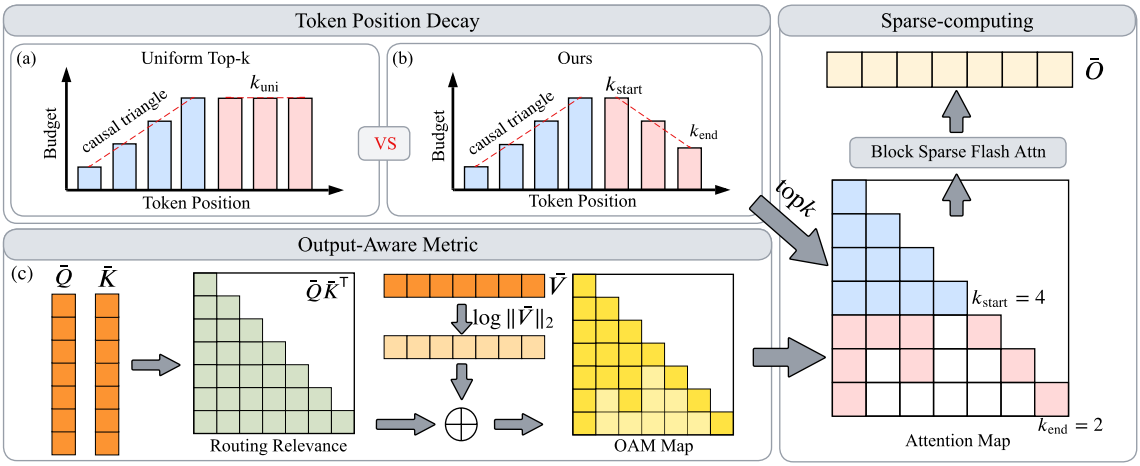

Figure 10: Architecture of the proposed Stem. (a) Uniform Sparse Attention: Illustrates the traditional method applying a fixed top-k budget across all token positions, often leading to information loss in initial tokens. (b) Token Position-Decay (TPD): Contrasts with (a) by allocating dynamic budgets, assigning higher retention stability to initial tokens ("recursive anchors") to preserve causal structure. (c) Output-Aware Metric (OAM): A selection metric combining attention scores with value vector magnitudes to minimize output approximation error.

Token Pruning for Multimodal Redundancy: A universal framework tailored for VLMs and Speech LLMs that reduces the effective sequence length by eliminating redundant visual and audio tokens.

4.1 Sparse Attention Framework

To mitigate the quadratic computational complexity associated with long-context LLMs, AngelSlim provides a comprehensive sparse attention framework. This framework is specifically optimized for the KV Cache Generation stage, enabling efficient prefill for sequences exceeding 100K tokens by exploiting the intrinsic sparsity within attention maps.

4.1.1 Overall Framework

A Versatile Library of Sparse Attention Patterns. We provide a versatile sparse attention library to accommodate the diverse attention behaviors observed across mainstream LLMs. The framework integrates several sophisticated strategies, primarily categorized into static and dynamic sparsity patterns. Static sparsity utilizes structured patterns based on attention distribution heuristics to maintain a fixed sparsity mask during inference. This primarily involves A-shape and Tri-shape configurations, and also supports Dilated and Strided attention to ensure a broad receptive field while significantly reducing the number of active tokens. Dynamic sparsity involves identifying necessary tokens on-the-fly during inference, which first performs pattern computation to locate sparse regions and then executes sparse attention kernels accordingly. Our framework supports a variety of advanced dynamic sparse algorithms, including MInference, xAttention, FlexPrefill, and Stem.

Decoupling Sparse Kernels from Model Architectures. We implement a strict decoupling between various sparse kernels and model architectures to simplify the deployment of diverse sparse algorithms. AngelSlim acts as a unified management layer that orchestrates various high-performance open-source kernels tailored for different sparse attention mechanisms. Through a training-free and metadata-driven configuration system, researchers can flexibly apply optimal sparsity settings to specific layers or heads for models like Qwen3, Llama3.1, and the Hunyuan-Dense-Model series. This design significantly accelerates the prefill stage, effectively reducing the time-to-first-token (TTFT) during long-context inference.

Automated Evaluation for Long-Context Capabilities. We integrate an automated evaluation pipeline that systematically benchmarks sparsified models on mainstream long-context suites. The system supports end-to-end evaluation on benchmarks such as LongBench for document understanding and RULER for testing retrieval robustness under high-ratio sparsity. By providing detailed metrics on sparsity ratio, time-to-first-token (TTFT), and task accuracy, this pipeline enables users to identify the most efficient configuration for large-scale deployment.

4.1.2 Algorithm

Stem: Rethinking Causal Information Flow. We introduce Stem, a plug-and-play, training-free sparsity module that prunes attention in a way that better matches causal information flow during long-context prefill. As illustrated in Fig. 10(a), a common baseline is uniform sparse attention: applying the same top-k budget to every query position, which often over-prunes the beginning of the sequence and discards information that many later tokens repeatedly depend on. Stem addresses this with two simple, report-friendly ideas shown in Fig. 10(b)(c). First, Token Position-Decay (TPD) (Fig. 10(b)) allocates a non-uniform budget across positions: early tokens are treated as "recursive anchors" and receive higher retention stability, while the budget gradually decays toward later tokens where redundancy is typically higher and aggressive sparsification is safer. Second, Output-Aware Metric (OAM) (Fig. 10(c)) changes how we decide what to keep: instead of selecting tokens purely by attention affinity, we also account for the effective contribution of the corresponding Value states, so tokens that look "high-score" but carry weak value signal are less likely to be selected, and tokens with meaningful value contribution are prioritized to reduce output distortion. Together, TPD provides a position-aware pruning schedule and OAM provides a contribution-aware selection rule, so Stem can preserve the critical early-sequence structure while still delivering large compute savings under high sparsity in long-context workloads.

Method	CC	FSL	MD1	MD2	SUM	SYN	AVG
Qwen3-8B							
Dense	19.09	63.10	11.23	15.06	20.63	62.92	32.01
MINF	18.91	61.87	10.78	14.46	20.26	55.32	30.27
FLEX	19.75	55.31	10.76	13.49	**21.18**	50.83	28.55
XATTN	**21.03**	**62.30**	**11.42**	14.64	20.47	52.92	30.46
Stem	19.43	61.84	11.22	**14.97**	20.21	**62.19**	**31.64**
Llama-3.1-8B-Instruct							
Dense	36.08	64.29	27.54	31.19	25.10	67.92	42.02
MINF	35.43	62.98	25.42	29.00	25.12	**68.41**	41.06
FLEX	34.65	59.51	17.17	21.62	24.46	59.10	36.09
XATTN	**37.23**	**63.61**	22.15	28.23	**25.30**	50.95	37.91
Stem	35.86	62.89	**26.33**	**30.53**	24.93	68.32	**41.48**
DeepSeek-V3.2 (trained sparse)							
DSA	**32.42**	41.85	**51.83**	48.30	**22.28**	60.35	42.84
DSA + Stem	32.02	**44.03**	51.67	**48.44**	21.97	**60.85**	**43.16**

Table 11: LongBench accuracy of Stem.

Effectiveness Across Dense and Trained-Sparse Backbones. Table 11 reports LongBench accuracy across task families, including code completion (CC), few-shot learning (FSL), multi-document QA (MD1 and MD2), summarization (SUM), and synthetic tasks (SYN), with AVG as the overall average. In training free deployment on dense backbones, Stem consistently achieves stronger overall accuracy than prior sparse baselines on both Qwen3 and Llama, showing that it can be inserted into existing models without retraining and remain reliable. Stem also transfers to models with trained sparse attention. On DeepSeek-V3.2 (Liu et al., 2024a), which uses DSA as its native sparse attention, applying Stem on top of DSA maintains comparable performance, indicating that Stem can refine token selection and remove residual redundancy without changing the trained model. Figure 11 further shows that Stem consistently reduces long-context prefill latency, since its metric computation remains lightweight and the position-decay schedule enables faster sparse execution.

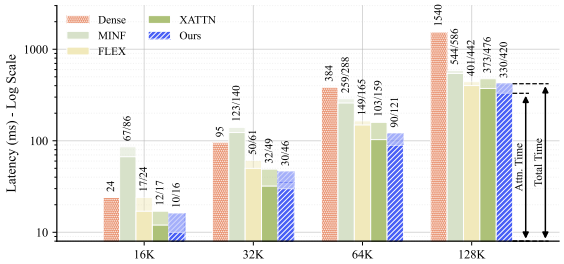

Figure 11: Latency comparison (ms) on Nvidia GPU. Results are reported as Attention Kernel Time/Total Time.

Other Sparse Attention Strategies. Beyond Stem, our framework provides comprehensive support for a wide range of common sparse attention Strategies, including FlexPrefill (Lai et al., 2025), MInference (Jiang et al., 2024), and XAttention (Xu et al., 2025). By integrating these techniques into a unified codebase, we streamline the experimental workflow and ensure fair comparisons among different sparsity paradigms.

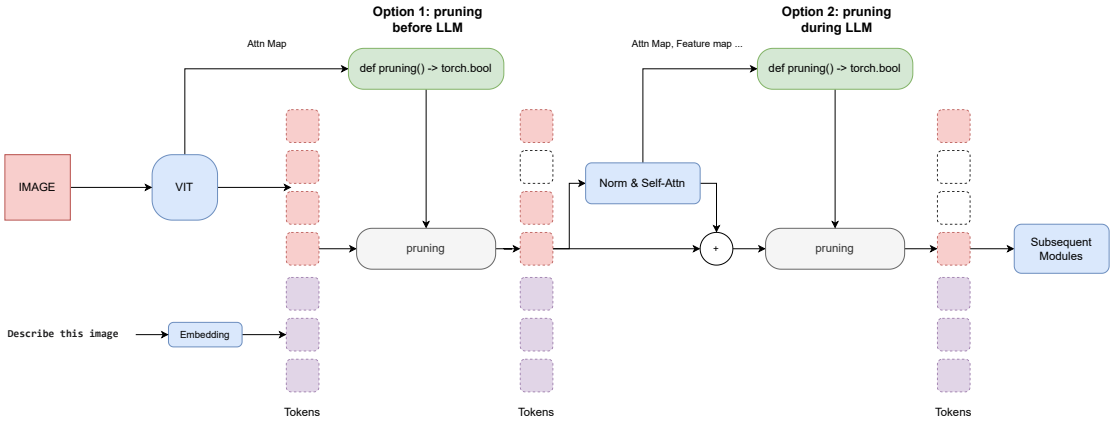

Figure 12: Architecture of the Universal Metadata-Driven Token Pruning Framework. The framework streamlines the pruning process by supporting two flexible compression schedules: Option 1 (Global Pruning) reduces the sequence length before entering the LLM by utilizing metadata from the Vision Tower (e.g., Attention Maps); Option 2 (Layer-wise Pruning) enables incremental sparsification between Transformer blocks based on intermediate states such as feature maps. The core design abstracts the pruning logic into a standardized interface (`def pruning() -> torch.bool`), effectively decoupling the algorithmic strategy from the underlying model architecture.

4.2 Token Pruning

In this subsection, we present our universal framework for token pruning and merging in multimodal settings, along with two novel algorithms tailored to different modalities: IDPruner for visual token pruning and Samp for audio token pruning and merging.

4.2.1 Framework

In this work, we present a universal metadata-driven framework designed for token pruning and merging in Multi-modal Large Language Models (MLLMs). As illustrated in Fig. 12, our architecture provides a standardized interface to simplify the overall token optimization process. This approach allows researchers to evaluate whether model performance is effectively maintained across various benchmarks while simultaneously tracking metrics such as theoretical computational speedup and inference latency.

Decoupling Pruning Strategies from Model Architectures. We implement a strict decoupling between pruning algorithms and specific model implementations to simplify the development cycle. Algorithm researchers only need to define the core pruning strategy within a standalone function that outputs a boolean mask based on runtime context. This design successfully isolates developers from the intricate details of the underlying model architectures. Once the mask is generated, the framework automatically handles essential downstream operations, such as slicing hidden states and synchronizing metadata—including attention masks, position embeddings, and KV cache position vectors. For strategies requiring intermediate information such as attention maps, the necessary data can be requested through YAML configurations. The framework dynamically captures these tensors during the forward pass and passes them as arguments to the pruning function.

Integrated End-to-End Evaluation Pipeline. We also provide a comprehensive solution that unifies algorithm development with systematic performance benchmarking. By integrating mainstream multi-modal evaluation suites directly into the compression engine, we have created a fully automated end-to-end pipeline. Through a unified interface, users can systematically evaluate optimized models across a suite of downstream tasks—such as TextVQA, MME, and DocVQA—at various pruning ratios. The system automatically generates detailed reports that present both task accuracy and theoretical computational speedup. This closed-loop design

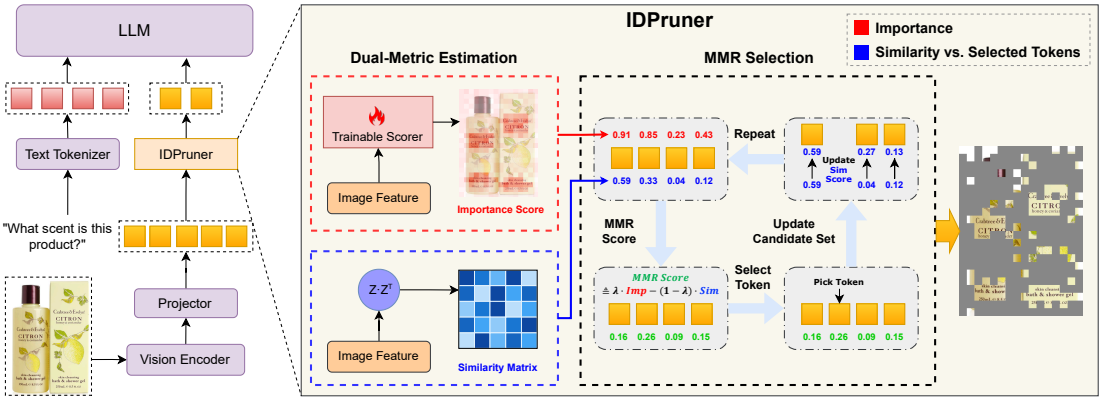

Figure 13: Architecture of the proposed IDPruner with importance-diversity optimization. By computing Importance Scores and a Similarity Matrix, IDPruner dynamically balances token saliency and semantic diversity through an MMR-based iterative selection process.

Method	AI2D *EM*	ChartQA *Relaxed*	DocVQA *Anls*	MMB^{CN} *Score*	MMB *Score*	MME *Score*	MMStar *Avg*	OCRBench *Acc*	POPE *Acc*	SQA *EM*	VQA^{Text} *EM*	Avg
Baseline	82.48	83.68	94.90	80.41	83.08	1702	61.88	85.30	87.80	88.45	82.74	100.0%
Retain 25% Tokens (75% Compression Ratio)												
FastV	75.68	68.20	81.20	73.20	76.12	1636	51.08	43.00	85.20	83.49	80.06	87.16%
VisionZip	77.40	67.20	71.48	76.12	78.78	1637	54.86	46.50	85.76	83.99	76.21	87.55%
HiPrune	77.49	68.60	73.52	76.03	78.09	1619	54.43	47.10	86.02	84.18	76.43	87.80%
VisionSelector	79.60	72.00	93.24	75.86	78.78	1688	55.78	72.50	86.74	85.08	80.39	94.22%
DivPrune	77.98	62.00	85.32	75.77	77.84	1650	52.97	58.40	85.88	83.94	75.88	89.26%
DART	74.35	60.80	78.90	73.88	76.72	1625	52.90	46.00	84.34	84.33	71.68	85.74%
VisPruner	77.62	68.04	77.39	75.69	78.87	1657	54.01	48.70	85.68	84.18	75.17	88.31%
SCOPE	78.92	71.20	85.40	77.75	79.38	1684	56.86	61.70	86.78	85.23	79.66	92.51%
IDPruner	80.51	74.32	93.16	76.63	79.73	1695	56.49	74.00	87.06	85.52	80.83	95.18%
Retain 10% Tokens (90% Compression Ratio)												
FastV	67.23	39.48	51.90	53.26	55.58	1332	38.02	24.10	76.31	79.28	72.59	68.07%
VisionZip	70.60	41.56	37.94	66.67	71.05	1462	45.19	23.40	81.06	83.24	61.06	71.84%
HiPrune	69.82	43.96	39.89	67.44	70.88	1438	45.04	23.70	80.70	82.65	62.51	72.22%
VisionSelector	74.81	62.68	87.00	68.99	71.65	1569	46.93	55.50	82.69	81.95	74.52	85.39%
DivPrune	70.11	41.36	66.20	69.42	72.16	1529	44.46	31.80	81.91	80.96	62.72	76.09%
DART	67.88	34.84	49.86	63.92	67.35	1451	42.93	24.30	79.70	80.96	54.06	69.80%
VisPruner	69.88	42.68	50.85	66.84	70.96	1442	44.14	24.40	81.03	81.11	59.66	72.60%
SCOPE	71.63	50.04	56.45	71.22	75.43	1608	48.74	34.10	84.10	82.25	70.61	79.35%
IDPruner	75.16	62.48	85.98	71.65	74.66	1618	47.48	53.90	85.43	82.80	74.43	86.47%

Table 12: Performance comparison of different pruning methods on Qwen-2.5-VL-7B-Instruct.

effectively reduces engineering overhead and accelerates the transition of advanced compression methods from theoretical research to practical deployment.

4.2.2 Visual Token Pruning Algorithms

IDPruner: A State-of-the-Art Pruning Method. We propose IDPruner, a novel strategy that reformulates visual token pruning as a re-ranking problem using the Maximal Marginal Relevance (MMR) algorithm. Unlike previous methods that rely on single metrics, IDPruner explicitly models the interplay between token importance and semantic diversity to achieve a Pareto-optimal balance. As illustrated in Figure 13, the algorithm iteratively selects tokens that maximize normalized saliency scores while minimizing redundancy relative to the already selected subset. Extensive experiments across diverse benchmarks and model architectures demonstrate that IDPruner achieves state-of-the-art performance, as shown in Table 12.

Other Pruning Strategies. Beyond IDPruner, our framework provides comprehensive support for a wide range of common pruning and merging methods, including VisionSelector (Zhu et al., 2025), FastV (Chen et al., 2024a), DivPrune (Alvar et al., 2025), DART (Wen et al., 2025), VisionZip (Yang et al., 2024), VisPruner (Zhang et al., 2025a), and SCOPE (Deng et al., 2025). The unified implementation of these diverse

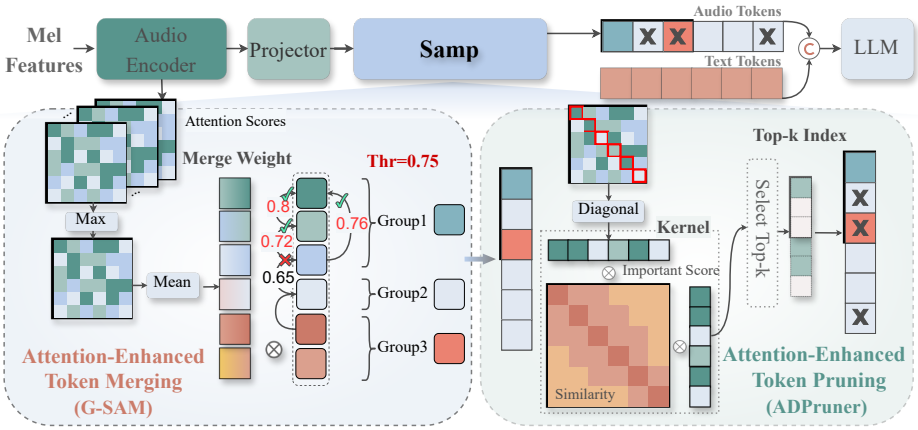

Figure 14: Architecture of the proposed **Samp** with adaptive merge–prune fusion. By introducing a similarity threshold, **Samp** dynamically balances token merging and pruning, according to speech temporal structures. Similar tokens are first adaptively merged to maximize information retention, followed by pruning to reduce redundancy and improve diversity under high compression rates.

methodologies enables researchers to perform systematic comparisons across different pruning paradigms with minimal effort.

4.2.3 Audio Token Pruning Algorithms

Samp: a plug-and-play, Similarity-Attention synergistically driven framework for joint token Merging and Pruning. Methods that integrate token pruning modules into LLM Transformer blocks often encounter compatibility bottlenecks, as highly optimized kernels like FlashAttention (Dao et al., 2022) restrict access to internal attention scores. To maximize hardware efficiency and ensure compatibility with optimized inference kernels, we position the Samp module before the LLM. In contrast to approaches that require deep integration into Transformer blocks, our method relies solely on attention scores from a single audio encoder layer, which significantly reduces architectural coupling and memory overhead. Tailored to the temporal dependencies and redundancy of speech tokens, we propose a two-part merging-pruning pipeline, as shown in Figure 14.

In the first merging stage, we group adjacent tokens based on intra-group global feature similarity and perform attention-guided weighted mergeing to collapse redundant segments while preserving critical semantic information. Given the audio features $\boldsymbol{H}_a \in \mathbb{R}^{N\times D}$ generated by the projection module preceding the LLM, we compute the pairwise cosine similarity across tokens to derive the similarity matrix $\boldsymbol{L}$. As illustrated in Fig. 14, we fix the merge similarity threshold λ and iterate through the audio token list $S = \{1, ..., N\}$ to sequentially incorporate multiple adjacent indices into a merged cluster. Let $s_i = \{k, ..., k + m - 1\}$ denote the set of audio tokens in the original merged cluster, where the cardinality of s_i is m. We then compute the average similarity between the next adjacent token (with index $k + m$) and the tokens in s_i , which is formally defined as:

$$l_{k+m,s_i} = \frac{1}{m}\sum_{t=k}^{k+m-1} \boldsymbol{L}_{k+m,t}, \qquad \boldsymbol{L}_{k+m,t} = \frac{\boldsymbol{H}_a^{k+m}\cdot \boldsymbol{H}_a^{t}}{\|\boldsymbol{H}_a^{k+m}\|\cdot\|\boldsymbol{H}_a^{t}\|}. \tag{8}$$

If $l_{k+m,s_i} \geq \lambda$, we incorporate the token indexed by $k+m$ into s_i ; otherwise, we initialize a new cluster $s_{i+1} = \{k + m\}$. The above procedure is repeated iteratively until all audio tokens are assigned to corresponding clusters. Ultimately, we yield K merged clusters, formally denoted as $S^* = \{s_i | i = 1, \ldots, K\}$. To maximize the discriminative importance information encoded by attention, we compute the maximum value across the head dimension, and then derive the important score $W_j \in \mathbb{R}^N$ of each audio token. Then, we aggregate the audio features of each subset s_i in the cluster set S^* (derived from the aforementioned clustering process) into a single token feature , ultimately yielding K compressed audio tokens denoted as $\tilde{\boldsymbol{H}}_a = \{\tilde{\boldsymbol{H}}_a^i | i = 1, ..., K\}$. and the corresponding index list set is updated to $\tilde{S} = \{1, ..., K\}$, in which:

$$\tilde{\boldsymbol{H}}_a^i = \frac{\sum^{|s_i|} \boldsymbol{W}_j \cdot \boldsymbol{H}_a^j}{\sum^{|s_i|} \boldsymbol{W}_j}, \qquad \boldsymbol{W}_j = \frac{1}{N}\sum_{n=0}^{N} \max_h \boldsymbol{A}_{h,n,j}. \tag{9}$$

Method	LibriSpeech				Fleurs		AISHELL-1	AISHELL-2	WenetSpeech		Average
	dev_clean	dev_other	test_clean	test_other	zh	en			test-meeting	test-net	
Qwen2-Audio	1.67	3.65	1.74	4.03	3.63	5.20	1.52	3.08	8.40	7.64	4.06
Retain 60% Tokens (40% Compression Ratio)											
VisionZip	7.31	10.35	7.08	10.10	8.02	6.85	6.99	13.85	17.88	23.75	11.22
VisPruner	7.42	9.74	7.20	9.69	7.00	8.39	6.91	9.67	13.37	14.07	9.35
CDPruner	4.22	6.05	4.18	6.53	4.88	7.17	2.70	4.62	12.29	10.90	6.35
A-ToMe	4.12	6.81	4.20	6.98	8.00	8.13	4.18	5.56	14.05	14.15	7.62
FastAdaSP	4.91	7.26	4.95	7.51	5.47	7.31	3.28	4.69	11.51	12.30	6.92
Samp	**2.59**	**5.00**	**2.72**	**5.02**	**4.37**	**5.94**	**2.69**	**4.42**	**11.05**	**10.11**	**5.39**
Kimi-Audio	1.23	2.39	1.38	2.45	2.87	4.92	0.61	2.57	6.33	5.39	3.01
Retain 60% Tokens (40% Compression Ratio)											
VisionZip	6.35	7.94	5.93	7.71	7.63	8.91	6.84	10.01	14.65	19.51	9.55
VisPruner	5.36	6.90	5.07	6.75	5.73	7.82	4.36	7.65	13.70	18.56	8.19
CDPruner	5.70	6.96	5.44	6.78	6.73	8.66	5.07	9.21	14.95	18.41	8.79
A-ToMe	4.13	6.04	3.92	5.87	6.26	7.57	3.58	6.88	12.97	14.99	7.22
FastAdaSP	4.49	6.15	4.28	6.02	4.27	7.64	2.22	5.67	12.62	14.75	6.81
Samp	**2.32**	**3.67**	**2.49**	**3.75**	**4.37**	**6.26**	**2.05**	**3.99**	**11.47**	**12.35**	**5.27**
GLM-ASR-Nano	2.14	4.05	2.18	4.53	3.44	4.11	2.47	3.48	8.43	6.65	4.15
Retain 70% Tokens (30% Compression Ratio)											
VisionZip	9.18	12.75	8.27	13.27	8.17	8.08	13.9	20.55	19.36	20.45	13.40
VisPruner	9.22	12.33	8.34	12.13	6.95	8.33	11.68	14.57	18.11	16.56	11.82
CDPruner	5.38	7.75	5.28	7.76	4.99	5.93	4.88	5.94	14.21	11.70	7.38
A-ToMe	4.55	8.04	4.66	7.83	5.08	5.77	5.40	5.95	13.16	12.08	7.25
FastAdaSP	6.45	10.34	6.50	10.41	4.34	6.70	6.42	7.02	16.10	14.10	8.84
Samp	**3.41**	**5.53**	**3.43**	**5.55**	**3.76**	**4.83**	**3.53**	**4.52**	**12.07**	**9.60**	**5.62**

Table 13: Performance comparison of different pruning methods on ASR dense tasks (Qwen2-audio / Kimi-Audio / GLM-ASR)

where N denotes the length of the audio token sequence, H stands for the number of attention heads, and attention scores $\boldsymbol{A} \in \mathbb{R}^{H\times N\times N}$, $j \in s_i$, and $\tilde{\boldsymbol{H}}_a^i$ denotes the merged feature corresponding to the cluster subset s_i. In the next pruning Stage, we implement a pruning kernel that integrates both attention and similarity information for diversity-driven pruning. Specifically, we weight the original kernel matrix by the derived importance scores to construct a novel conditional kernel matrix:

$$\hat{\boldsymbol{L}} = diag\left(\hat{\boldsymbol{A}}\right) \cdot \boldsymbol{L} \cdot diag\left(\hat{\boldsymbol{A}}\right), \qquad \hat{\boldsymbol{A}} = \frac{1}{H}\sum_{i=0}^{H} \boldsymbol{A}_{h,n,j}. \tag{10}$$

We then obtain the optimal subset via MAP inference (Chen et al., 2024c). By incorporating a similarity thresholding mechanism, **Samp** adaptively calibrates the ratio between merging and pruning for each individual sample, achieving a dynamic equilibrium between compression aggressiveness and task-specific accuracy.

Other Pruning Strategies. Beyond **Samp**, our framework also provides pure merging methods like A-ToMe (Li et al., 2023) and FastAdaSP (Lu et al., 2024), pure pruning methods like VisPruner (Zhang et al., 2025a) and CDPruner (Zhang et al., 2025b), and a hybrid approach which is VisionZip (Yang et al., 2024). By comparing these methods, the advantages of our proposed approach can be clearly demonstrated, for example, by its performance on the ASR task in Table 13.

5 Conclusion and Future Works

In this technical report, we introduced **AngelSlim**, a unified and comprehensive framework designed to surmount the "Inference Wall" by synergizing disparate model compression paradigms. By integrating weight-level quantization, structural sparsity, speculative decoding, and multimodal token pruning into a cohesive, hardware-aware manifold, AngelSlim provides a robust pipeline for scaling large foundation models. Our empirical evaluations demonstrate that AngelSlim achieves state-of-the-art performance across all optimization axes. Specifically, our ultra-low-bit quantization strategies, including the 2-bit QAT and ternary quantization methods, achieve accuracy parity with higher-precision formats while delivering a $4\times$ speedup on edge devices. Furthermore, the Eagle3 speculative decoding framework and our training-free sparse attention library significantly enhance inference throughput and long-context efficiency. By decoupling algorithmic strategies from low-level system implementations, AngelSlim empowers both researchers and practitioners to deploy

LLMs and MLMs at a fraction of the original computational cost without compromising reasoning integrity. Ultimately, this toolkit serves as a critical infrastructure for the democratization of large-scale AI, bridging the gap between theoretical algorithmic innovation and practical industrial application.

6 Contributions and Acknowledgments

AngelSlim are the result of the collective efforts of all members of Tencent HY AngelSlim team. Below, we will list contributors.

Contributors (Ordered by the last name): Rui Cen, QiangQiang Hu, Hong Huang, Hong Liu, Song Liu, Xin Luo, Lin Niu, Yifan Tan, Decheng Wu, Linchuan Xie, Rubing Yang, Guanghua Yu, Jianchen Zhu

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9392–9401, 2025. URL https://api.semanticscholar.org/CorpusID:276775957.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

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.

Siyu Cao, Hangting Chen, Peng Chen, Yiji Cheng, Yutao Cui, Xinchi Deng, Ying Dong, Kipper Gong, Tianpeng Gu, Xiusen Gu, et al. Hunyuanimage 3.0 technical report. *arXiv preprint arXiv:2509.23951*, 2025.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Tianqi Chen, Zhe Li, Weixiang Xu, Zeyu Zhu, Dong Li, Lu Tian, Emad Barsoum, Peisong Wang, and Jian Cheng. Ternaryllm: Ternarized large language model. *arXiv preprint arXiv:2406.07177*, 2024b.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024c.

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

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Jinhong Deng, Wen Li, Joey Tianyi Zhou, and Yang He. Scope: Saliency-coverage oriented token pruning for efficient multimodel llms. *ArXiv*, abs/2510.24214, 2025. URL https://api.semanticscholar.org/CorpusID:282401430.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

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

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Hong Huang and Dapeng Wu. Quaff: Quantized parameter-efficient fine-tuning under outlier spatial stability hypothesis. *arXiv preprint arXiv:2505.14742*, 2025.

Hong Huang, Decheng Wu, Rui Cen, Guanghua Yu, Zonghang Li, Kai Liu, Jianchen Zhu, Peng Chen, Xue Liu, and Dapeng Wu. Tequila: Trapping-free ternary quantization for large language models. *arXiv preprint arXiv:2509.23809*, 2025.

Hong Huang, Decheng Wu, Qiangqiang Hu, Guanghua Yu, Jinhai Yang, Jianchen Zhu, Xue Liu, and Dapeng Wu. Sherry: Hardware-efficient 1.25-bit ternary quantization via fine-grained sparsification. *arXiv preprint arXiv:2601.07892*, 2026.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36:62991–63010, 2023.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Ayush Kaushal, Tejas Vaidhya, Arnab Kumar Mondal, Tejas Pandey, Aaryan Bhagat, and Irina Rish. Surprising effectiveness of pretraining ternary language model at scale. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11260–11285, 2024.

Yuang Li, Yu Wu, Jinyu Li, and Shujie Liu. Accelerating transducers through adjacent token merging. *arXiv preprint arXiv:2306.16009*, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025a.

Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. Quantization meets reasoning: Exploring llm low-bit quantization degradation for mathematical reasoning. *arXiv preprint arXiv:2501.03035*, 2025b.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

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

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, et al. Paretoq: Scaling laws in extremely low-bit llm quantization. *arXiv preprint arXiv:2502.02631*, 2025.

Yichen Lu, Jiaqi Song, Chao-Han Huck Yang, and Shinji Watanabe. Fastadasp: Multitask-adapted efficient inference for large speech language model. *arXiv preprint arXiv:2410.03007*, 2024.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Shuming Ma, Hongyu Wang, Shaohan Huang, Xingxing Zhang, Ying Hu, Ting Song, Yan Xia, and Furu Wei. Bitnet b1. 58 2b4t technical report. *arXiv preprint arXiv:2504.12285*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Tencent HY Team. Hunyuan-0.5b-instruct, 2025a. URL https://huggingface.co/tencent/Hunyuan-0.5B-Instruct.

Tencent HY Team. Hunyuan-1.8b-instruct, 2025b. URL https://huggingface.co/tencent/Hunyuan-1.8B-Instruct.

Tencent HY Team. Hunyuan-a13b, 2025c. URL https://github.com/Tencent-Hunyuan/Hunyuan-A13B/blob/main/report/Hunyuan_A13B_Technical_Report.pdf.

Tencent HY Team. Hunyuan-dense-model, 2025d. URL https://huggingface.co/collections/tencent/hunyuan-dense-model.

Tencent HY Team. Hunyuan-1.8b-instruct-gptq-int4, 2025e. URL https://huggingface.co/tencent/Hunyuan-1.8B-Instruct-GPTQ-Int4.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jinheng Wang, Hansong Zhou, Ting Song, Shijie Cao, Yan Xia, Ting Cao, Jianyu Wei, Shuming Ma, Hongyu Wang, and Furu Wei. Bitnet. cpp: Efficient edge inference for ternary llms. *arXiv preprint arXiv:2502.11880*, 2025.

Jianyu Wei, Shijie Cao, Ting Cao, Lingxiao Ma, Lei Wang, Yanyong Zhang, and Mao Yang. T-mac: Cpu renaissance via table lookup for low-bit llm deployment on edge. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 278–292, 2025.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for important tokens in multimodal language models: Duplication matters more. *ArXiv*, abs/2502.11494, 2025. URL https://api.semanticscholar.org/CorpusID:276408079.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

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*, 2025a.

Rubing Yang, Huajun Bai, Song Liu, Guanghua Yu, Runzhi Fan, Yanbin Dang, Jiejing Zhang, Kai Liu, Jianchen Zhu, and Peng Chen. Specexit: Accelerating large reasoning model via speculative exit. *arXiv preprint arXiv:2509.24248*, 2025b.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models, 2024. URL https://arxiv.org/abs/2412.04467.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025.

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

Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms, 2025a. URL https://arxiv.org/abs/2412.01818.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond attention or similarity: Maximizing conditional diversity for token pruning in mllms. *arXiv preprint arXiv:2506.10967*, 2025b.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Jiaying Zhu, Yurui Zhu, Xin Lu, Wenrui Yan, Dong Li, Kunlin Liu, Xueyang Fu, and Zheng-Jun Zha. Visionselector: End-to-end learnable visual token compression for efficient multimodal llms, 2025. URL https://arxiv.org/abs/2510.16598.