OPENPANGU EMBEDDED-1B: EFFICIENT SMALL LANGUAGE MODELS FOR ASCEND EDGE DEVICES

openPangu Team, Huawei

openPangu@huawei.com

ABSTRACT

The rapid advancement of large language models (LLMs) has significantly advanced the capabilities of artificial intelligence across various domains. However, their enormous computational demands and high energy consumption render them unsuitable for resource-constrained edge environments. To bridge this gap, we propose openPangu Embedded-1B, an efficient yet effective language model optimized specifically for Ascend edge devices. The model features a hardware-aware architecture that balances the trade-off between accuracy and inference efficiency. Our training pipeline consists of multi-stage pre-training on a diverse multilingual corpus, curriculum-based supervised fine-tuning (SFT), offline on-policy knowledge distillation, and reinforcement learning with a multi-source reward mechanism aimed at enhancing mathematical and coding proficiency. The resulting instruction-tuned model, openPangu Embedded-1B-V1.1, achieves state-of-the-art performance among billion-parameter models, demonstrating strong generalization under strict hardware constraints while maintaining competitive accuracy across a variety of tasks. This work provides a practical and efficient solution for developing high-performance language models on Ascend edge devices. The code and models are available at https://gitcode.com/ascend-tribe/openpangu-embedded-1b-v1.1.

1 Introduction

"A sparrow may be small, but it has all its vital organs." — Chinese proverb

Large Language Models (LLMs) have transformed the landscape of artificial intelligence by leveraging self-attention mechanisms and massive-scale pre-training to capture hierarchical linguistic patterns, semantic relationships, and cross-domain knowledge [2]. Open-sourced models such as LLaMA [34], DeepSeek [24], Qwen [6, 38] and openPangu [8, 31] have demonstrated exceptional performance in complex tasks including text generation, reasoning, and multilingual understanding. However, their success hinges on enormous parameter counts (*e.g.*, DeepSeek-V3 [24] has 671B parameters in total) and extensive computational resources, which limit their deployment in latency-sensitive or resource-constrained environments. The prohibitive energy consumption and hardware requirements of LLMs have raised critical challenges for real-world applications, particularly in scenarios demanding on-device processing, privacy preservation, or low-latency responsiveness.

To address these limitations, a paradigm shift has emerged toward developing efficient small language models (SLMs) for edge devices and resource-constrained platforms. The growing ubiquity of smartphones, IoT sensors, and embedded systems has heightened the demand for lightweight models that retain competitive performance under stringent memory, power, and computational constraints. Recent advances in SLM architecture design focus on creating more inherently efficient model structures. This includes the use of hybrid model architectures [12], and scalable training strategies [20] that reduce inference-time complexity without compromising representational capacity. Furthermore, significant progress continues in model compression techniques. Methods such as knowledge distillation (e.g., GKD [3]), structured pruning (e.g., SparseGPT [15] and Pangu Light [7]), and model quantization (e.g., SmoothQuant [36] and CBQ [11]) are widely adopted. These techniques enable dramatic reductions in model size, memory footprint, and computational requirements, facilitating practical on-device deployment.

In this work, we present openPangu Embedded-1B, a small language model specifically developed for efficient inference on Ascend edge devices. The model architecture is designed for high-speed on-device deployment, incorporating structural choices that reduce latency and maximize hardware utilization. The resulted architecture runs efficiently on Ascend edge devices such as Ascend Atlas 200I A2. The large-scale pre-training on about 10T data is conducted on Ascend Atlas 800T A2 clusters. It encompasses multiple stages: pre-training on a diverse multilingual and multi-domain corpus, followed by an annealing phase with high-quality data to enhance convergence and stability. Supervised fine-tuning (SFT) is conducted using a curriculum that progresses from step-by-step to fast-response examples, strengthening general instruction-following capability. The instruct model after SFT is named as openPangu Embedded-1B-V1.

Building upon the V1 version, we further enhance model performance through advanced post-training strategies. Specifically, we employ offline on-policy knowledge distillation from a larger teacher model that shares the same tokenizer, enabling efficient logit-level knowledge transfer. Additionally, reinforcement learning is applied using a multi-source reward system specifically optimized to boost performance in mathematical reasoning and code generation tasks. The resulting instruction-tuned model, named openPangu Embedded-1B-V1.1, benefits from this comprehensive training pipeline and achieves state-of-the-art accuracy among billion-scale instruction models.

2 Model Architecture and Pretraining

In this section we will introduce the architecture of openPangu Embedded-1B and the details of its pretraining stage.

2.1 Model Architecture

The openPangu Embedded-1B is a 1-billion-parameter, decoder-only Transformer model [35] architected for efficient inference on the Ascend Atlas 200I A2 platform. Its design results from a deliberate hardware-software co-design process, which tailors the model's structure to the specific computational and memory characteristics of the target hardware.

While studies such as [30, 43] suggest that deeper models can enhance capacity for a given parameter count, this approach often compromises inference speed on resource-constrained edge devices. To address this trade-off, our primary objective is to strike an optimal balance between model performance and inference efficiency. The resulting hyperparameters of openPangu Embedded-1B are detailed in Table 1. Notably, the hidden size and FFN intermediate size are specifically chosen to align with the hardware's architecture. This ensures efficient utilization of the Ascend compute units, effectively preventing computational idling and reducing overhead from memory misalignment.

To validate our hardware-aware design, we benchmark the inference latency of openPangu Embedded-1B against other models of a similar scale. As shown in Table 1, openPangu Embedded-1B achieves a Time to First Token (TTFT) of 1834 ms and a Time per Output Token (TPOT) of 156 ms. Given that openPangu Embedded-1B maintains a level of accuracy comparable to these models [38], its superior inference speed demonstrates the significant advantages of our co-design methodology in achieving state-of-the-art performance on the target edge platform.

Table 1: Inference efficiency analysis on Ascend Atlas 200I A2. The latency is measured under the setting of FP16 format, batch size of 1, input sequence length of 1024 and output sequence length of 256. TTFT: Time To First Token. TPOT: Time Per Output Token. Avg Acc is the averaged accuracy in Table 4.

Model	Layers	Vocab. size	Hidden size	Heads (Q / KV)	FFN intermediate size	TTFT (ms)	TPOT (ms)	Avg Acc
Qwen2.5-1.5B	28	151936	1536	12 / 2	8960	2171	178	55.10
Qwen3-1.7B	28	151936	2048	16 / 8	6144	3243	213	63.69
openPangu Embedded-1B	26	153376	1536	12 / 6	6144	1834	156	63.90

2.2 Pre-training data

The pre-training corpus of openPangu Embedded-1B contains high-quality and diverse 10.4T tokens produced by our domain-aware tokenizer with a vocabulary size of 153,376 tokens, as described in more detail

in [41]. This corpus is derived from various sources, including web content, books, multilingual, code, STEM (Science, Technology, Engineering, and Mathematics), industrial domains, reasoning, and synthetically generated data. Therefore, the pre-training corpus is classified into seven major categories: General English and Chinese, Multi-lingual, Industrial, Instruction, STEM, and Code.

Training Phases Table 2 shows that the pre-training process is structured into three sequential phases: the *general* phase, the *reasoning* phase, and the *annealing* phase. These phases are designed to progressively establish foundational knowledge and linguistic proficiency, enhance reasoning skills, and further refine both knowledge integration and behavioral adaptability, respectively. The amount of data used in *general* phase is 8T, including two distinct subphases: 4T with a 4K sequence length and the other 4T with an 8K sequence length. The *reasoning* phase takes 1.5T tokens with an 8K sequence length, and the *annealing* phase takes 0.9T tokens with a 32K sequence length. The sequence length gradually increases from 4K to 32K during the three phases.

We annotate the data with quality and difficulty labels, and implement a curriculum learning-inspired sampling strategy for reasoning examples, particularly from the STEM, instruction-style, and code dataset. This approach dynamically progresses through three distinct phases, presenting increasingly complex examples to optimize the learning trajectory. Overall, the pre-training data for openPangu Embedded-1B is meticulously curated to prioritize high quality, broad diversity, and minimal redundancy.

Table 2: Data recipe of openPangu Embedded-1B pre-training.

Dataset	General	Reasoning	Annealing
General English	45%	22%	22%
General Chinese	12%	12%	11%
Multi-lingual	10%	4%	3%
Industrial	7%	10%	8%
Instruction	2%	10%	20%
STEM	7%	20%	17%
Code	17%	22%	19%

3 Post-training Strategy

In this section, we present the post-training strategy of our openPangu Embedded-1B including Two-Stage Curriculum SFT, Offline On-policy Knowledge Distillation and Reinforcement Learning, as shown in Fig 1.

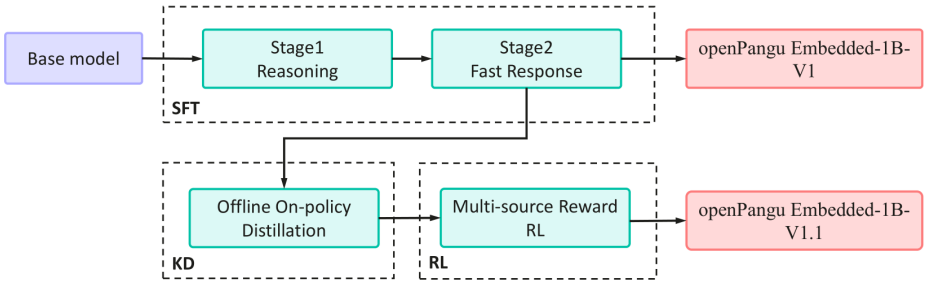


Figure 1: An illustration of the openPangu Embedded-1B post-training pipeline. The pipeline consists of three primary stages: Two-Stage Curriculum SFT, Offline On-policy Knowledge Distillation, and Reinforcement Learning.

3.1 Post-training Data

For post-training data, core principles of high quality, diversity, and complexity are prioritized, with a reasoning-centered design to enhance the model's capabilities and generalization as well. The initial data pool integrates multi-source data: open-source instruction datasets, real-world industrial queries (*e.g.,* finance/healthcare scenarios), and synthetic problems derived from pre-training corpora. The post-training data are split into two key subsets: reasoning tasks (advanced STEM with multi-step computation, code

generation requiring symbolic manipulation, logical inference) and non-reasoning tasks (general QA, text analysis, long-context understanding, semantic classification, tool use, and agent), with a 3:1 sampling ratio adopted.

To ensure data quality, a rigorous two-stage processing pipeline is constructed: 1) Prior filtering: we leverage our models to annotate data with attributes (subcategory, question type, answer verifiability, difficulty metrics like reasoning hops) to filter unqualified samples; 2) Diversity maintenance: We use N-gram-based MinHash-LSH to eliminate near-duplicates, followed by the ZIP algorithm (guided by entropy [40]) to select samples by prioritizing low compression ratios (higher diversity) and minimizing similarity to existing entries, to yield a pattern-rich dataset. The overall post-training dataset emphasizes reasoning tasks as well as dataset diversity, thus strengthening the model's ability to avoid superficial pattern matching, while non-reasoning tasks support basic language competence, collectively enhancing performance across both specialized reasoning tasks and general abilities.

3.2 Two-Stage Curriculum SFT

In the SFT phase, we propose the Two-Stage Curriculum SFT, a method operationalized through a difficulty-aware two-stage curriculum. The design is inspired by a fundamental insight from cognitive science, which posits that proficient general reasoning is a prerequisite for its subsequent, more intuitive execution.

- **Stage 1 (Reasoning):** The initial stage focuses on building the cognitive foundation. The model trains on an extensive corpus of reasoning-enhanced data, which features explicit, step-by-step reasoning chains. This phase instills deep inferential abilities and prevents reliance on shallow heuristics.
- **Stage 2 (Fast Response):** Subsequently, the curriculum is converted to "fast response" data, consisting of conventional prompt-response pairs that omit intermediate reasoning. This encourages the model to implicitly activate and leverage its acquired cognitive framework, focusing on generating concise and accurate final outputs.

This training method effectively activates the model's reasoning abilities while enhancing its general capabilities, improving model precision, and maintaining fast-response ability.

3.3 Offline On-policy Knowledge Distillation

On-policy distillation [3] has been proven to be an effective model compression technique. It leverages the outputs of a larger, more powerful teacher model to guide a smaller student model, thereby enhancing the student's accuracy and generalization capabilities. This process enables compact student models to achieve superior performance.

To decouple the generation of teacher model guidance data from the training process of the student model, and to allow independent optimization of the training data and process, we design an *offline version of on-policy distillation*. Unlike the online version, which cannot dynamically adjust and optimize the quality of the generated logits, the offline version allows for selecting high-quality data during the preprocessing stage to generate teacher model guidance information. Compared to the standard SFT process, our method introduces only an additional Knowledge Distillation (KD) loss term. This design simplifies the implementation process, ensuring ease of use and straightforward operation, while enhancing the flexibility and controllability of the system.

Our method is systematically structured into two distinct phases: an offline data preparation phase followed by a training phase. The offline data preparation phase includes two key components: Student-driven Response Generation and Teacher's Token-Level Logits Prediction:

- **Student-driven Response Generation:** The process commences with our SFT-trained student model, which performs inference on the queries from the original training dataset. This generates an intermediate on-policy dataset, D_s, where each sample consists of a query and the corresponding student-generated response. This initial step, which we refer to as *distillation by student*, ensures that the subsequent teacher guidance is grounded in the student's own output distribution.
- **Teacher's Token-Level Logits Prediction:** Following response generation, we employ a powerful teacher model to annotate each sequence in D_s with token-level guidance. For each position n in a student-generated sequence, the teacher model is conditioned on the prefix of the preceding

$n-1$ tokens to predict the distribution for the n-th token. Crucially, rather than performing greedy decoding, we preserve the teacher's full predictive distribution by recording the logits for the top-k most probable tokens. This constrained conditioning strategy is the cornerstone of our approach. By compelling the teacher to generate guidance from a probabilistic space already accessible to the student, we effectively minimize the intrinsic distributional divergence between the two models. This facilitates a more stable and efficient knowledge transfer.

Training with a Composite Loss Function In the training phase, the student model is optimized using a composite loss function that synergistically combines standard supervised learning with knowledge distillation. The total loss L_{total} is formulated as the weighted sum of the Cross-Entropy (CE) loss L_{CE} and the knowledge distillation loss L_{KD}:

$$L_{total} = L_{CE} + \lambda_{KD} \cdot L_{KD}$$

where λ is a scalar hyperparameter that is a weighting coefficient. It meticulously balances the influence of the direct supervised objective (L_{CE}) and the teacher's distributional guidance (L_{KD}).

The core of the distillation process is the minimization of the KL divergence, denoted $D_{KL}(P \parallel Q)$, is an asymmetric measure that quantifies how a probability distribution Q (from the student) differs from a reference probability distribution P (from the teacher). For discrete distributions over a vocabulary of classes C, it is defined as:

$$D_{KL}(P \parallel Q) = \sum_{c \in C} P(c) \log \frac{P(c)}{Q(c)}$$

By minimizing the KL divergence, the student's output distribution (Q) is trained to approximate the soft-target distribution (P) provided by the teacher. A lower divergence value signifies that the student has successfully learned to mimic the teacher's predictive patterns, effectively internalizing the knowledge transferred during distillation.

3.4 Reinforcement Learning Training Policy

We employ the same reinforcement learning (RL) training recipe as openPangu Embedded-7B [8]. Here we summarize the key components. First, we utilize the Group Relative Policy Optimization (GRPO) algorithm [28], a standard approach in the post-training phase. A significant issue emerges when every response to a particular prompt is assigned the same reward. This scenario leads to a zero normalized advantage, which can reduce the GRPO objective to a basic behavior cloning loss and hinder the policy's exploratory capabilities. To address this, we have implemented a "Zero-Advantage-Mask" mechanism, which effectively disregards samples with a zero advantage during the loss computation. As a result, the policy is exclusively updated using data that provides a distinct learning signal (i.e., a non-zero advantage), fostering more effective learning and robust exploration.

To provide nuanced and task-specific guidance, our multi-source reward system dynamically routes prompts to specialized evaluators based on task characteristics, assigning correctness-based rewards for verifiable domains like math and code through a hybrid of rule-based verifiers and a multi-stage execution process. In contrast, for open-domain creative tasks, it employs a normalized, LLM-based preference model to emulate human judgment, and simultaneously incorporates orthogonal auxiliary rewards from a format validator and a repetition penalty to uphold structural integrity and conciseness without distorting the primary learning objectives.

4 Experiments

We commence by detailing the implementation of openPangu Embedded-1B. This is followed by a comprehensive evaluation of both its base and instruction-tuned versions on widely-used benchmarks. Finally, we present and discuss the key findings from our ablation studies.

4.1 Main Results

In this section, we evaluate openPangu Embedded-1B on both reasoning and normal language tasks.

4.1.1 Pre-training Evaluation

Training Setup We pre-train openPangu Embedded-1B from scratch using the AdamW [25] optimizer with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The pre-training curriculum is designed to build foundational capabilities progressively and is divided into two main stages: *1)* The first stage consists of three phases: general ability training with a sequence length of 4K, followed by general ability and reasoning ability training, both at a sequence length of 8K. Across these phases, we use a global batch size of 16 million tokens and a cosine learning rate schedule that anneals from 5×10^{-3} to 1×10^{-5}; *2)* The second stage is a final annealing phase focused on enhancing long-sequence capabilities. For this, we increase the sequence length to 32K. The learning rate again follows a cosine schedule, while the batch size remains constant. All training tasks are conducted on Ascend Atlas 800T A2 clusters.

Baselines & Benchmarks We compare openPangu Embedded-1B against several strong, similarly-sized open-source baselines to situate its performance within the current state-of-the-art landscape. These include Qwen3 [38], Gemma3 [32], and Llama3.2 [14]. Our evaluation spans a comprehensive suite of 11 benchmarks covering four key capability areas: English, Chinese, Mathematics, and Code:

- *English Language*. We use MMLU [16] (Massive Multitask Language Understanding) to assess broad general knowledge across 57 subjects; BigBenchHard [29] (BBH), a collection of challenging BIG-Bench tasks requiring multi-step reasoning; HellaSwag [42], which tests commonsense NLI by asking models to predict the most plausible continuation of a given scenario; and WinoGrande [1], a large-scale benchmark for commonsense reasoning via pronoun resolution problems.
- *Chinese Language*. To evaluate Chinese capabilities, we employ CMMLU [22], a benchmark designed to test college-level knowledge and reasoning in a Chinese context; C-Eval [21], a comprehensive suite of multiple-choice questions from 52 diverse disciplines in China; and CLUEWSC [37], a Winograd Schema Challenge task for Chinese commonsense reasoning.
- *Mathematical Reasoning*. Mathematical abilities are tested using GSM8K [10], a dataset of grade-school math word problems that require multi-step arithmetic reasoning; and MATH [17], a benchmark of challenging problems from high school math competitions.
- *Code Generation & Reasoning*. For coding, we use MBPP [5] (Mostly Basic Python Problems), which contains entry-level programming problems; and HumanEval [9], which tasks models with generating correct Python code from docstrings.

For base models, the majority of our evaluations employ few-shot settings, with a minority using zero-shot prompts. We use exact matching to evaluate most benchmarks with gold answers, while employing execution-based verification for code generation tasks.

Evaluation Results As shown in Table 3, openPangu Embedded-1B establishes a new performance benchmark for 1B-parameter models, consistently outperforming its peers and significantly closing the gap with larger models like Qwen3-1.7B. The model demonstrates strong, well-balanced capabilities in both English and Chinese, achieving high scores on general knowledge benchmarks such as MMLU (54.17) and CMMLU (57.25). In code generation, the model shows specialized strength, achieving the top score of 34.15 on HumanEval, outperforming all other models in the comparison. This combination of balanced bilingualism, exceptional mathematical prowess, and proficient coding ability makes openPangu Embedded-1B a highly parameter-efficient model, delivering advanced capabilities suitable for resource-constrained environments like edge and embedded systems.

4.1.2 Post-training Evaluation

SFT Training Setup The SFT phase of openPangu Embedded-1B uses a two-stage fine-tuning approach, each stage running for 10 epochs. For overall training stability, we employ the AdamW optimizer with a weight decay of 0.1 and apply gradient clipping at a threshold of 1.0. We set the maximum sequence length to 32K to maximize computational efficiency, packing multiple samples into each sequence.

The first stage focuses on complex reasoning, using a global batch size of 4 million tokens. Its learning rate follows a cosine schedule with a 200-iteration warmup, annealing from a peak of 2×10^{-5} down to 2×10^{-6}. The second stage targets open-ended generation tasks, using a smaller global batch size of 2 million tokens. The learning rate for this stage also follows a cosine schedule, decaying from 1×10^{-5} to 1×10^{-6}.

Table 3: Base model comparison between openPangu Embedded-1B and other representative base models across a diverse set of benchmarks for evaluating language and reasoning skills. If the original paper reports the results, we present the results from the original paper (marked with asterisks *); otherwise, we list our reproduced results.

	Benchmark (Metric)	**# Shots**	**Qwen3**	**Gemma3**	**Llama3.2**	**Qwen3**	**openPangu Embedded**
	# Total Params	-	1.7B	1B	1B	0.6B	1B
English	MMLU	5-shot	62.63*	26.14	32.12	52.81*	54.17
	BigBenchHard	3-shot	54.47*	28.18	29.68	41.47*	40.68
	HellaSwag	10-shot	57.71	24.72	24.72	37.15	50.74
	WinoGrande	5-shot	56.12	50.51	50.91	50.51	52.01
Chinese	CMMLU	5-shot	66.49	25.23	26.49	52.65	57.25
	C-Eval	5-shot	64.68	25.50	27.36	56.36	58.59
	CLUEWSC	5-shot	53.07	49.69	50.00	50.31	53.07
Math	GSM8K	8-shot	82.11	14.56	22.97	70.96	67.78
	MATH	4-shot	43.50*	15.22	17.60	32.44*	45.70
Code	MBPP	3-shot	55.40*	8.20	24.00	36.60*	34.80
	HumanEval	3-shot	32.93	8.54	12.80	24.39	34.15
	Average		57.19	25.14	28.97	45.97	49.90

RL Training Setup In RL training, we train openPangu Embedded-1B using GRPO with the clip higher mechanics and a batch size 256 for 6 epochs. Each input is sampled 8 times at a maximum sequence length of 32K with sampling parameters set to temperature = 1.0, top-k = -1, and top-p = 1.0. The clipping range is restricted to 0.8∼1.28.

The RL training data is mined from a pool of 30k queries, sourced from datasets like orz-math [19], NuminaMath [23], etc. Concretely, based on the KD model, we produce multiple responses for each query and select those with a pass rate between 0 and 1 as training data, yielding 7k data for RL training. Notably, the RL training data exclusively consists of math problems, we observe that training on math data also improves reasoning abilities in other domains, such as coding.

Optimization is performed with the AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We adopt a cosine annealing learning rate schedule, decaying from 1×10^{-6} to 5×10^{-7} within the first 100 iterations. To ensure stability, gradient norms are clipped at 0.5. In addition, a KL penalty with coefficient 0.001 is applied to regularize the learned policy against the reference model, preventing distributional drift. The reward function is designed to encourage format adherence and correctness: non-conforming outputs are penalized with -1, conforming but incorrect outputs receive -0.5, and correct answers are rewarded with +1.

Baselines & Benchmarks The openPangu Embedded-1B family is compared against several prominent open-source models within a similar parameter class to ensure a relevant and competitive analysis. The compared baselines include Qwen3 (1.7B and 0.6B) [38], Qwen2.5(1.5B) [39], Gemma3 (1B) [32], Llama3.2 (1B) [14] and MiniCPM4 (0.5B) [33]. Including the larger Qwen3-1.7B model serves as a critical point of comparison, allowing for an evaluation of the parameter efficiency of the proposed approach. Performance is measured using standard metrics appropriate for each benchmark. Accuracy (Acc) is used for tasks with single correct answers, such as MMLU [16] and GSM8K [10]. The F1 Score is employed for tasks like DROP [13], which require a balance of precision and recall in text extraction. For code generation tasks like MBPP [5] and HumanEval [9], Pass@1 is used, which measures the percentage of problems for which a correct solution is generated in a single attempt.

To ensure a fair and comprehensive evaluation, the evaluation suite is meticulously curated to probe a wide range of cognitive abilities, categorized into four primary domains:

- *General tasks*: Fundamental language understanding, multi-domain knowledge, and chinese language proficiency are assessed using established benchmarks. These include MMLU for broad, multi-disciplinary knowledge; CMMLU [22] and C-Eval [21] for comprehensive Chinese language evaluation; IF-Eval [44] for instruction-following fidelity; and CLUEWSC [37] for commonsense

Table 4: Instruct model (non-thinking) comparison between openPangu Embedded-1B and other representative models across a diverse set of benchmarks for evaluating language and reasoning skills. **Bold** values represent the best results in each line. If the original paper reports the results, we present the results from the original paper (marked with asterisks *); otherwise, we list our reproduced results. To ensure the stability of the results, we evaluate the openPangu Embedded-1B-V1.1 three times using vLLM-Ascend [4] and take the average as the final result.

	Benchmark (Metric)	**Qwen3**	**Qwen2.5**	**Gemma3**	**Llama3.2**	**Qwen3**	**MiniCPM4**	**openPangu Embedded** V1	**openPangu Embedded** V1.1
	# Total Params	1.7B	1.5B	1B	1B	0.6B	0.5B	1B	1B
General	MMLU(Acc)	63.37	52.84	37.49	32.19	44.24	55.55*	60.72	**65.08**
	CMMLU (Acc)	61.22	53.98	31.57	10.81	42.94	**65.22***	51.99	55.65
	C-Eval (Acc)	61.00*	59.30	32.49	32.08	42.60*	**66.11***	60.98	63.02
	IF-Eval (Prompt Strict)	**68.20***	42.50*	51.57	39.37	54.50*	50.28	56.56	55.51
	CLUEWSC(Acc)	77.36	74.59	50.20	52.36	50.31	49.90	68.55	**78.55**
Math	GSM8K (Acc)	77.03	73.20*	57.16	36.39	59.29	52.08*	66.72	**82.76**
	MATH-500 (Acc)	73.00*	46.60	39.40	18.20	55.20*	29.60*	52.00	**81.83**
Reasoning	DROP (F1)	**61.21**	48.44	30.98	45.23	34.69	30.07	50.31	52.82
	GPQA-Diamond (Pass@1)	28.60*	29.80*	19.20*	29.29	22.90*	28.28	33.84	**41.70**
Code	MBPP (Pass@1)	60.70	**63.20***	58.75	40.08	46.69	59.14*	54.09	59.31
	HumanEval (Pass@1)	**68.90**	61.60*	40.24	29.88	40.85	46.34*	56.71	66.66
	Average	63.69	55.10	40.82	33.26	44.93	48.42	55.68	**63.90**

reasoning. These benchmarks test a model's core knowledge base and ability to apply it in varied contexts.

- *Mathematics*: To evaluate complex, multi-step quantitative reasoning, the evaluation employs GSM8K and MATH-500 [17]. These benchmarks require numerical computation and the critical ability to translate natural language problems into logical steps, testing the depth of a model's reasoning capacity.

- *Reasoning*: The models' capacity for complex reasoning and information extraction is tested using DROP [13], which measures reading comprehension intertwined with arithmetic reasoning, and GPQA-Diamond [26]. This question-answering dataset that probes deep, domain-specific reasoning in physics and chemistry.

- *Code Generation*: Proficiency in programming is evaluated using MBPP [5] and HumanEval [9]. These benchmarks assess the ability to synthesize correct and functional code from natural language docstrings, a key skill for practical applications.

Decoding Strategy. For openPangu Embedded-1B-V1, the inference is conducted using a greedy decoding strategy with a decode length of 8K. The subsequent model, openPangu Embedded-1B-V1.1, results from applying knowledge distillation and RL. The RL training process necessitates a less restrictive approach to explore the potential token selections fully. The decoding method for openPangu Embedded-1B-V1.1 is therefore set to sampling to maintain consistency between training and inference. Specifically, the parameters are configured as follows: do_sample is enabled, top-k is set to -1 (disabled), top-p is 1.0, temperature is 0.7, and the decode length is 28K.

Evaluation Results The empirical results (summarized in Table 4) unequivocally establish the superior performance of the openPangu Embedded model family, with the openPangu Embedded-1B-V1.1 setting a new state-of-the-art for models in the 1B parameter class. A salient finding is its aggregate performance, achieving an average score of 63.90, which is on par with the larger Qwen3-1.7B model (63.69). This demonstrates exceptional parameter efficiency, suggesting that advanced training and alignment methodologies can be more impactful than simply scaling model size. The model's most profound advantage lies in mathematical and complex reasoning, where it achieves leading scores on GSM8K (82.76) and MATH-500 (81.83). This strength is complemented by robust general knowledge capabilities, securing top positions on benchmarks like CLUEWSC (78.55) and MMLU (65.08), highlighting a robust and well-rounded bilingual foundation.

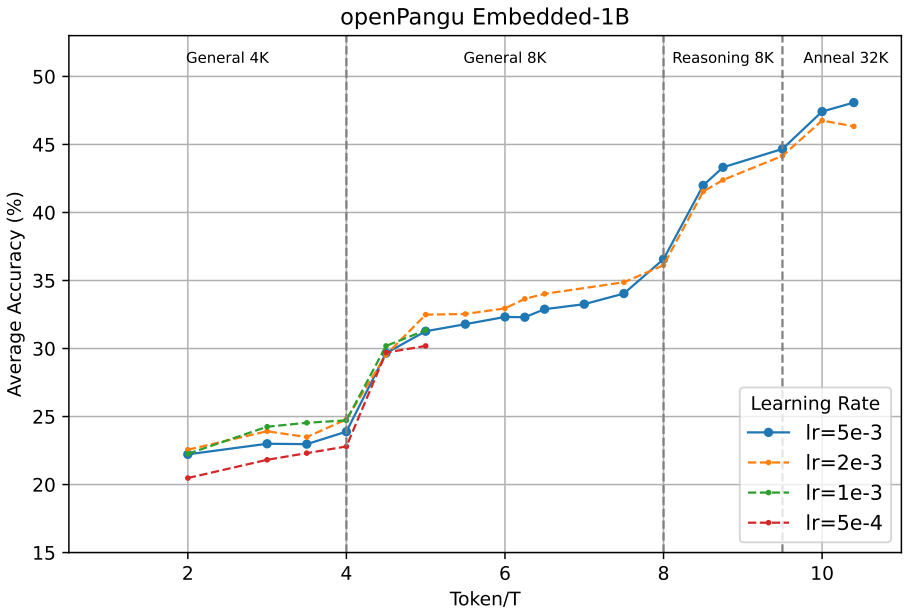


Figure 2: Evolution of pre-training accuracy with various learning rates. The evaluation metric is the average few-shot accuracy across eight datasets: MMLU(5-shot), CMMLU(5-shot), CEval(5-shot), BBH(3-shot), GSM8K(8-shot), MATH(3-shot), MBPP(3-shot), and HumanEval(3-shot). All results are generated using greedy decoding.

4.2 Ablation Studies

4.2.1 Pretraining Learning Rate

We conduct a meticulous tuning process and ablation study for the pre-training learning rate. To determine the optimal learning rate, we experiment with a range of learning rates: 5×10^{-3}, 2×10^{-3}, 1×10^{-3}, and 5×10^{-4}, to observe their respective impacts on model accuracy.

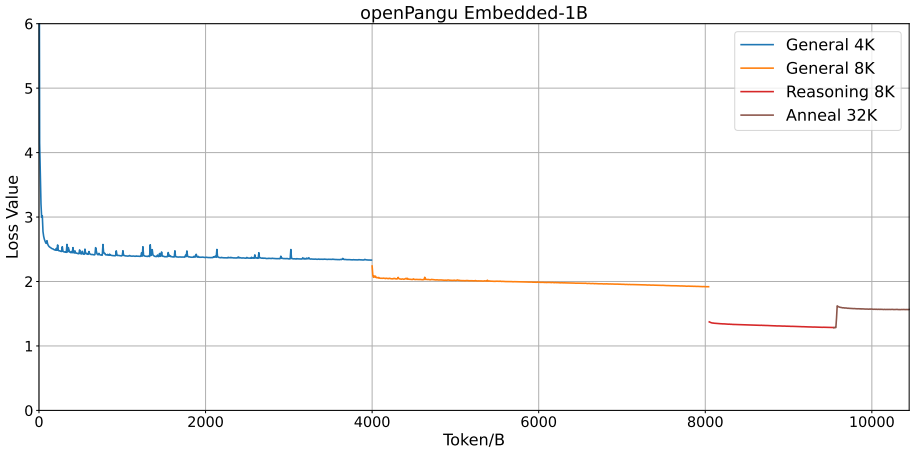
As illustrated in Fig 2, while the model trained with a learning rate of 5×10^{-3} exhibits lower accuracy during the initial training phases, it surpasses the other models upon completion of the general capability training stage. Notably, its accuracy continues to improve significantly during the annealing phase.

Fig 3 illustrates the training loss trajectory, which follows a consistent downward trend throughout the training process. A key observation relates to the impact of the learning rate on initial training stability. Specifically, the model trained with a higher learning rate of 5×10^{-3} exhibits pronounced loss spikes in the early stages. In contrast, models trained with lower learning rates display considerably smoother loss profiles. Crucially, as substantiated by our final results (Fig 2), this initial volatility has a negligible impact on the final model accuracy. This finding suggests that for this architecture, early-stage loss spikes are a tolerable artifact of an aggressive learning rate and do not indicate poor final performance.

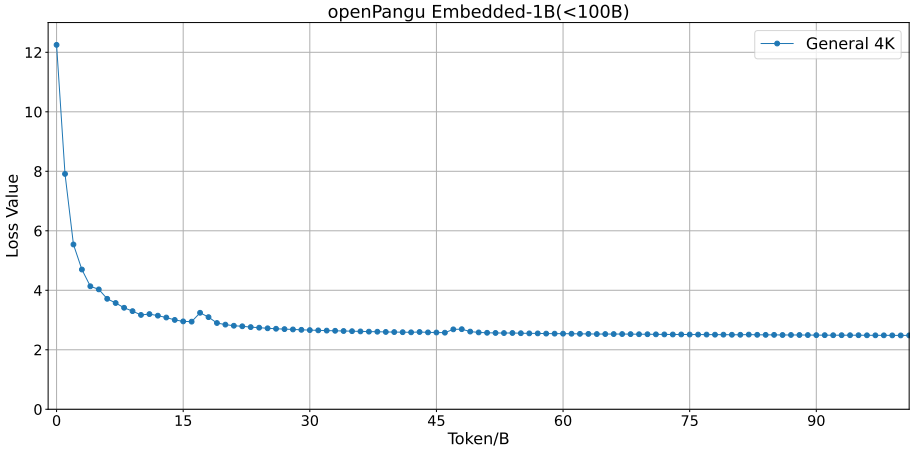
4.2.2 SFT

To optimize the SFT process, we conduct a series of ablation studies to investigate whether a multi-stage training curriculum can enhance the model's "fast thinking" or intuitive response capabilities. Specifically, we seek to determine whether first fine-tuning on deliberative reasoning tasks before fine-tuning on rapid-response data yields superior performance. We evaluate three distinct training methods:

- **One-Stage Curriculum (Direct Fast Response).** In this single-stage approach, the model is directly fine-tuned using only the fast response dataset. This baseline measures the efficacy of training exclusively on target-domain data without any preparatory learning phases.
- **Two-Stage Curriculum (Reasoning-to-Fast, without CoT).** This method includes a two-stage curriculum. The model is first trained on a Reasoning dataset from which the intermediate reasoning



(a) Loss trajectory for the full pre-training phase.



(b) Loss trajectory during the initial <100B tokens of the pre-training phase.

Figure 3: Pretraining loss trajectory for openPangu Embedded-1B with a learning rate of 5×10^{-3}. **(a)** Loss trajectory for the full pre-training phase, where the y-axis is clipped to a maximum value of 6 for improved readability. The pre-training process is divided into four stages: General 4K, General 8K, Reasoning 8K, and Annealing 32K. A consistent downward trend in loss is observed within each stage. Note that the baseline loss during the Reasoning phase is comparatively lower due to the characteristics of its training data. **(b)** Loss trajectory during the initial <100B tokens of the pre-training phase (from-scratch). The initial loss value is 12.25, and a rapid decrease in loss is observed during this early stage.

steps (i.e., the "thought process") have been explicitly removed. Following this, the model is fine-tuned on the fast response dataset. This approach tests the benefit of a sequential training regimen on datasets with different characteristics, without explicitly teaching the model to reason.

- **Two-Stage Curriculum (Reasoning-to-Fast, with CoT).** openPangu Embedded-1B is also trained in two stages. However, in the initial stage, it is fine-tuned on the complete Reasoning dataset, which crucially retains the detailed reasoning chains and thought processes. This stage is designed to instill deliberative reasoning capabilities before fine-tuning for rapid-response generation.

Table 5: Accuracy comparison of SFT methods with greedy decoding. All SFT models are fine-tuned from a pre-trained model with a learning rate of 2e-3 and a decode length of 8K.

Method	General					Math		Reasoning		Code		AVG
	MMLU	CMMLU	C-Eval	IF-Eval	CLUEWSC	GSM8K	MATH-500	DROP	GPQA-Diamond	MBPP	HumanEval	
Direct Fast-Response	60.71	**54.07**	58.50	57.12	57.89	62.32	44.20	34.68	31.82	50.19	53.66	51.38
Reasoning (w/o CoT) → Fast	**61.35**	50.19	59.47	**58.60**	**69.67**	64.44	47.20	42.06	**37.37**	50.97	**57.32**	54.42
Reasoning (w/ CoT) → Fast	60.72	51.99	**60.98**	56.56	68.55	**66.72**	**52.00**	**50.31**	33.84	**54.09**	56.71	**55.68**

The experimental outcomes highlight the efficacy of our proposed two-stage curriculum. Specifically, the Reasoning (w/o CoT)-to-Fast approach outperforms the direct Fast-Response method, yielding a 3.04% gain in accuracy. This suggests that a multi-stage approach, even without explicit "thought process", helps the model adapt more effectively by learning from a different data distribution before specializing.

Most significantly, the Reasoning (w/ CoT)-to-Fast approach achieves the highest accuracy of 55.68%, surpassing both other configurations. This result strongly supports our hypothesis. We equip the model with foundational logical and problem-solving skills by exposing it to data that includes explicit reasoning paths. This initial Reasoning phase acts as a form of cognitive scaffolding. When the model is subsequently fine-tuned on the "fast-response" data, it is not merely memorizing input-output pairs. However, it can leverage its previously acquired reasoning abilities to generate more robust and accurate responses. This demonstrates the profound value of a curriculum that prioritizes the development of underlying reasoning skills before optimizing for rapid, intuitive task completion.

4.2.3 Knowledge Distillation

We conduct a systematic ablation study on knowledge distillation to further enhance model performance. Our analysis focuses on four key dimensions: (i) the weighting of the distillation loss term, (ii) the effect of the top-k value during decoding, (iii) the timing of the distillation phase, and (iv) the choice of distillation strategy. This investigation aims to elucidate each factor's relative importance and identify the most effective configuration.

Knowledge Distillation Loss Weight. To optimize the balance between the standard cross-entropy loss and the distillation loss, we conduct experiments with different values of the KD loss weight (λ_{KD}), ranging from 0.5 to 1.0. To accelerate the experimental cycle, we conduct distillation experiments based on an intermediate version of the pre-trained model (learning rate: 2×10^{-3}), which has undergone only a single stage (Direct Fast Response) of SFT. This approach introduces a new Stage 2 to perform distillation guided by labels, with the distillation learning rate kept consistent with that used in SFT. This systematic comparison shows setting $\lambda_{KD} = 0.9$ yields the best overall performance, as shown in Table 6.

Table 6: Effect of KD loss weight on the average benchmark performance of stage2 distillation by label. The evaluation metric is the average zero-shot accuracy across eight benchmarks: MMLU, CMMLU, CEval, BBH, GSM8K, MATH, MBPP, and HumanEval. All models use greedy decoding with a decode length of 8K.

λ_{KD}	0.5	0.6	0.7	0.8	**0.9**	1.0
Average	53.71	53.80	53.63	54.09	**55.00**	54.14

Top-k Value Effect. To explore the impact of the number of top tokens used in the distillation process, we experiment with two values for the top-k parameter: 5 and 10. By adjusting the value of k, we aim to investigate how the number of candidate logits influences model performance. The experimental setup is consistent with that described in the *Knowledge Distillation Loss Weight* section. We observe that training time remains almost identical across different k values. In terms of performance, increasing k consistently

improves accuracy: top-k=10 outperforms top-k=5 by 0.43%, as shown in Table 7. Hence, we adopt $\lambda_{KD} = 0.9$ and top-k=10 by default unless otherwise specified.

Table 7: Effect of top-k value on the average benchmark performance of stage2 distillation by label. The evaluation metric is the average zero-shot accuracy across eight benchmarks: MMLU, CMMLU, CEval, BBH, GSM8K, MATH, MBPP, and HumanEval. All models use greedy decoding with a decode length of 8K, and the KD loss weight is fixed at $\lambda_{KD} = 0.9$.

Top-k	5	**10**
Average Performance	54.57	**55.00**

Distillation Phase Timing. We further investigate the optimal timing of the distillation process to achieve the best final performance. To prioritize model accuracy, we adjust the experimental setup. The following distillation experiments are conducted on a model that is based on the final pre-trained version (with a learning rate of 5×10^{-3}) and has undergone a two-stage SFT (reasoning-to-fast, with CoT) process. We examine two approaches.

- **Stage 2 Distillation:** Replacing the second stage of our existing SFT process with a distillation phase.
- **Stage 3 Distillation:** Appending a dedicated third stage for distillation, initialized from the checkpoint of the completed Stage 2.

Based on the findings from the *Knowledge Distillation Loss Weight* and *Top-K Value Effect* section, we adopt the optimal configuration with top-k=10 and λ_{KD}=0.9 for the following experiments. These experiments employ logits-based distillation conditioned on ground-truth labels. As shown in Table 8, replacing Stage 2 with distillation yields a 2.42% improvement. However, extending training with an additional Stage 3 delivers a slightly larger gain of 2.51%.

The results suggest that retaining the original SFT Stage 2 is essential, as it equips the student model with a strong task-specific foundation. Once this foundation is established, a subsequent distillation stage can more effectively align the student's representations with the teacher's output distribution. Thus, treating distillation as a complementary final step, rather than a substitute for core training, proves more effective in unlocking the model's full potential.

Table 8: Ablation study on knowledge distillation. "Distillation by Label" means teacher logits are conditioned on ground-truth labels, with the teacher's topk-10 predicted logits serving as auxiliary guidance, whereas "Distillation by Student" means teacher logits are conditioned on the student's own predictions to encourage self-consistency. All models use greedy decoding with a decode length of 8K.

Method	General					Math		Reasoning		Code		AVG
	MMLU	CMMLU	C-Eval	IF-Eval	CLUEWSC	GSM8K	MATH-500	DROP	GPQA-Diamond	MBPP	HumanEval	
Stage2 SFT	63.21	53.10	58.51	56.38	76.95	70.89	56.20	28.73	43.43	52.53	59.15	56.28
Stage2 Distillation by Label	61.38	55.12	60.14	56.93	75.41	72.78	64.60	**43.01**	42.93	51.36	61.59	58.66
Stage3 Distillation by Label	62.68	53.97	58.83	**59.70**	77.56	72.10	65.20	30.24	**46.46**	57.20	62.80	58.79
Stage3 Distillation by Student	**65.91**	**56.17**	**64.30**	59.33	**79.51**	**76.72**	**71.40**	40.99	44.95	**58.75**	**67.07**	**62.28**

Distillation Strategy. We conduct a key ablation study to compare two distillation strategies. Specifically, The first is the conventional approach (distillation by label) [18, 27], where teacher logits are conditioned on ground-truth labels. The second is offline onpolicy distillation strategy(distillation by student), where the student model first generates its own response, and the teacher then provides target logits based on that output. As shown in Table 8, the results demonstrate the clear superiority of offline onpolicy distillation(distillation by student), which achieves a substantial performance increase of 3.49%.

This finding highlights the benefit of aligning the teacher's guidance with the student's output space. By conditioning distillation on the student's response, the teacher provides corrective and refining signals on a distribution immediately relevant to the student's current state. This alignment minimizes the distributional mismatch between the two models, creating a more stable learning signal and facilitating more effective knowledge transfer.

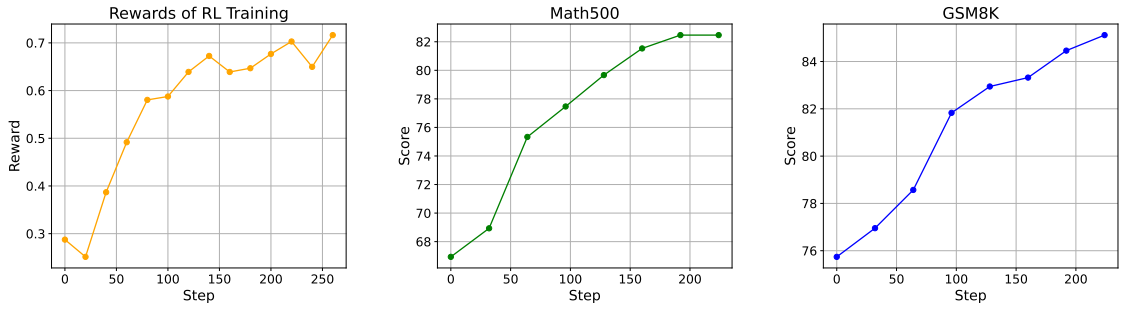


Figure 4: RL training performance for mathematical tasks. These curves illustrate that mathematical ability can steadily improve with RL training.

4.3 RL Training Process

The reward curve during the training process is shown in Fig.4, where we also append the test curves for Math500 and GSM8K. The test curves for other tasks are presented in Fig.5. Overall, the reward steadily improved during the RL training, resulting in a substantial boost in mathematical reasoning ability, while maintaining performance across other domains.

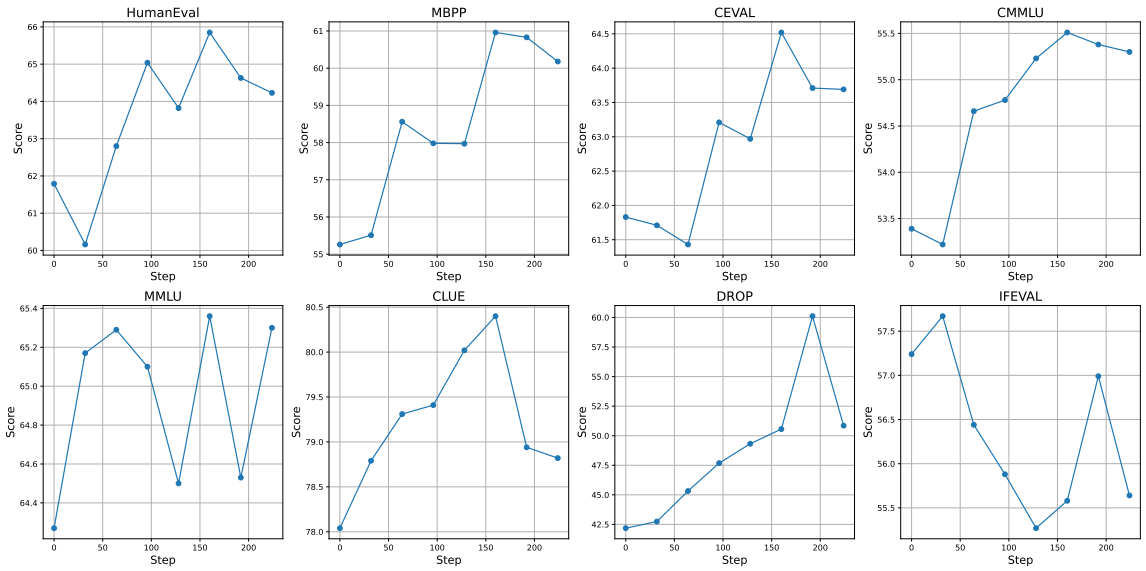


Figure 5: RL training performance for other tasks. These curves demonstrate that other domain abilities are well maintained during RL training.

5 Conclusion and Discussion

This work presents openPangu Embedded-1B, a highly efficient small language model optimized for deployment on Ascend edge devices. By integrating a hardware-aware architecture with a comprehensive multi-stage training pipeline including large-scale pre-training, curriculum-based SFT, offline on-policy knowledge distillation, and multi-source reward reinforcement learning, we have developed a compact yet powerful model that achieves state-of-the-art performance among models with around 1 billion parameters. All the training and deployment are conducted on Ascend hardware. Extensive experiments demonstrate that openPangu Embedded-1B excels in mathematical reasoning, code generation, and multilingual understanding while maintaining superior inference efficiency on Ascend edge hardware. This provides a practical and scal-

able solution for bridging high-performance AI and edge deployment, paving the way for advanced on-device intelligence. Future work will focus on expanding capabilities and optimizing for broader edge scenarios.

References

[1] Winogrande: An adversarial winograd schema challenge at scale. 2019.

[2] 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.

[3] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

[4] Ascend. vllm-ascend. https://github.com/vllm-project/vllm-ascend, 2025.

[5] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[6] 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.

[7] Hanting Chen, Jiarui Qin, Jialong Guo, Tao Yuan, Yichun Yin, Huiling Zhen, Yasheng Wang, Jinpeng Li, Xiaojun Meng, Meng Zhang, et al. Pangu light: Weight re-initialization for pruning and accelerating llms. *arXiv preprint arXiv:2505.20155*, 2025.

[8] Hanting Chen, Yasheng Wang, Kai Han, Dong Li, Lin Li, Zhenni Bi, Jinpeng Li, Haoyu Wang, Fei Mi, Mingjian Zhu, et al. Pangu embedded: An efficient dual-system llm reasoner with metacognition. *arXiv preprint arXiv:2505.22375*, 2025.

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura 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, Mo Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. 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. *ArXiv*, abs/2107.03374, 2021.

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[11] Xin Ding, Xiaoyu Liu, Zhijun Tu, Yun Zhang, Wei Li, Jie Hu, Hanting Chen, Yehui Tang, Zhiwei Xiong, Baoqun Yin, et al. Cbq: Cross-block quantization for large language models. In *ICLR*, 2025.

[12] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. Hymba: A hybrid-head architecture for small language models. *arXiv preprint arXiv:2411.13676*, 2024.

[13] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[14] 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 preprint arXiv:2407.21783*, 2024.

[15] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR, 2023.

[16] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[17] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[19] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

[20] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[21] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Fanchao Qi, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *ArXiv*, abs/2305.08322, 2023.

[22] 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. *arXiv preprint arXiv:2306.09212*, 2023.

[23] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

[24] 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*, 2024.

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

[26] 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.

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[28] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[29] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[30] Yehui Tang, Kai Han, Fangcheng Liu, Yunsheng Ni, Yuchuan Tian, Zheyuan Bai, Yi-Qi Hu, Sichao Liu, Shangling Jui, and Yunhe Wang. Pangu-π pro: Rethinking optimization and architecture for tiny language models. In *Forty-first International Conference on Machine Learning*, 2024.

[31] Yehui Tang, Yichun Yin, Yaoyuan Wang, Hang Zhou, Yu Pan, Wei Guo, Ziyang Zhang, Miao Rang, Fangcheng Liu, Naifu Zhang, et al. Pangu ultra moe: How to train your big moe on ascend npus. *arXiv preprint arXiv:2505.04519*, 2025.

[32] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[33] MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, et al. Minicpm4: Ultra-efficient llms on end devices. *arXiv preprint arXiv:2506.07900*, 2025.

[34] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[35] 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.

[36] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

[37] Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*, 2020.

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

[39] Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.

[40] Mingjia Yin, Chuhan Wu, Yufei Wang, Hao Wang, Wei Guo, Yasheng Wang, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. Entropy law: The story behind data compression and llm performance, 2024.

[41] Yichun Yin, Wenyong Huang, Kaikai Song, Yehui Tang, Xueyu Wu, Wei Guo, Peng Guo, Yaoyuan Wang, Xiaojun Meng, Yasheng Wang, Dong Li, Can Chen, Dandan Tu, Yin Li, Fisher Yu, Ruiming Tang, Yunhe Wang, Baojun Wang, Bin Wang, Bo Wang, Boxiao Liu, Changzheng Zhang, Duyu Tang, Fei Mi, Hui Jin, Jiansheng Wei, Jiarui Qin, Jinpeng Li, Jun Zhao, Liqun Deng, Lin Li, Minghui Xu, Naifu Zhang, Nianzu Zheng, Qiang Li, Rongju Ruan, Shengjun Cheng, Tianyu Guo, Wei He, Wei Li, Weiwen Liu, Wulong Liu, Xinyi Dai, Yonghan Dong, Yu Pan, Yue Li, Yufei Wang, Yujun Li, Yunsheng Ni, Zhe Liu, Zhenhe Zhang, and Zhicheng Liu. Pangu ultra: Pushing the limits of dense large language models on ascend npus, 2025.

[42] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[43] Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

[44] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Inference Efficiency Analysis on Ascend Atlas 300I DUO

Table 9 presents a comparative analysis of inference efficiency conducted on a single Ascend Atlas 300I DUO single chip. The experiments were performed under the FP16 precision, with a batch size of 1, an input sequence length of 1024, and an output sequence length of 256.

The results demonstrate that the openPangu Embedded-1B model achieves excellent performance in TTFT, requiring only 149 ms. This outperforms models of similar parameter scales such as Qwen3-0.6B (173 ms) and Qwen3-1.7B (247 ms), and is even lower than the fewer-layer Llama3.2-1B model (151 ms). In terms of TPOT, openPangu Embedded-1B achieves 62 ms, which is comparable to Qwen3-0.6B but significantly better than the larger-parameter Qwen3-1.7B (76 ms). It is worth noting that although Llama3.2-1B performs best in TPOT (41 ms), its model architecture employs a shallower architecture. Considering the findings in [30] that deeper layers with the same parameter count are more conducive to improving model accuracy, we adopt a deeper architecture in openPangu Embedded-1B. From the results in Table 4, openPangu Embedded-1B achieves much better performance over Llama3.2-1B. It shows that openPangu Embedded-1B achieves high performance while maintaining inference efficiency.

Table 9: Inference efficiency analysis on Ascend Atlas 300I DUO single chip. The latency is measured under the setting of FP16 format, batch size of 1, input sequence length of 1024 and output sequence length of 256. TTFT: Time To First Token. TPOT: Time Per Output Token. Avg Acc is the averaged accuracy in Table 4.

Model	**Layers**	**Vocab. size**	**Hidden size**	**Heads (Q / KV)**	**FFN intermediate size**	**TTFT (ms)**	**TPOT (ms)**	**Avg Acc**
Qwen3-0.6B	28	151936	1024	16 / 8	3072	173	62	44.91
Llama3.2-1B	16	128256	2048	16 / 8	8192	151	41	33.26
Qwen3-1.7B	28	151936	2048	16 / 8	6144	247	76	63.69
openPangu Embedded-1B	26	153376	1536	12 / 6	6144	149	62	63.90

B Response Examples

Case Study: Model Outputs on a Chinese Traffic Regulation Question

Question

驾驶机动车在高速公路上行驶，遇到能见度小于 50 米的大雾天气时，应该怎么做?

A）继续以 120 公里/小时的速度行驶，并开启远光灯。

B）将车速降至 20 公里/小时以下，并从最近的出口尽快驶离高速公路。

C）开启危险报警闪光灯，并就地停车等待大雾散去。

D）将车速降至 60 公里/小时，并与前车保持 100 米以上的距离。

Model Responses

openPangu Embedded-1B-V1.1

在高速公路上遇到能见度小于 50 米的大雾天气时，驾驶员必须采取以下安全措施：
正确做法是：
B）将车速降至 20 公里/小时以下，并从最近的出口尽快驶离高速公路。

解析：

1. **能见度极低时的风险**：能见度小于 50 米意味着高速公路上危险，驾驶员需要迅速降低车速以确保安全。
2. **车速限制**：根据《道路交通安全法》及相关规定，能见度小于 150 米时应开启雾灯（或近光灯）、保持车距，并必要时开启双闪灯。能见度小于 50 米时，车速应降至 **20 公里/小时以下**（或更低），以最大限度减少风险。
3. **驶离高速公路**：若无法安全通过出口或无法保持车距，应尽快驶离高速公路，避免在能见度极低的情况下长时间停留。
4. **其他选项分析**：
   - **A）继续以 120 公里/小时并开远光灯**：远光灯会增加后方车辆的眩目风险，且 120 公里/小时在能见度极低时仍为危险驾驶速度。
   - **C）开启双闪灯并就地停车**：”就地停车”在能见度极低时可能导致二次事故，且高速公路无应急车道，强行停车风险极高。
   - **D）车速降至 60 公里/小时并保持 100 米距离**：60 公里/小时在能见度极低时仍属危险，且 100 米距离不满足安全要求（建议至少 150 米）。

法律依据：
根据《中华人民共和国道路交通安全法实施条例》第八十一条：

> 能见度小于 200 米时，车速不得超过每小时 100 公里，开启雾灯、近光灯；能见度小于 100 米时，车速不得超过每小时 60 公里，车速应在车道内行驶；**能见度小于 50 米时，车速不得超过每小时 20 公里，车速应在车道内行驶，开启雾灯、双闪灯，从最近出口驶离高速公路。**

结论：
选项 B 是唯一符合法律规定的正确做法，同时兼顾了安全性和可操作性。

gemma-3-1b-it

正确答案是 C) 开启危险报警闪光灯，并就地停车等待大雾散去。
以下是原因：
危险报警闪光灯的作用：在能见度低于 50 米的大雾天气，车辆的视线受阻，对驾驶员的安全性构成严重威胁。开启危险报警闪光灯可以提醒其他车辆，并发出警告，从而降低事故发生的可能性。
就地停车等待：在能见度低时，驾驶员需要立即采取行动，避免因视线不清而发生事故。就地停车，等待大雾散去，是安全行驶的最佳方式。
其他选项的错误：
A) 继续以 120 公里/小时的速度行驶，并开启远光灯：即使在能见度低的情况下，继续以高速度行驶会增加事故风险。
B) 将车速降至 20 公里/小时以下，并从最近的出口尽快驶离高速公路：降低车速并迅速离开高速公路会增加车辆的反应时间，可能导致事故。
D) 将车速降至 60 公里/小时，并与前车保持 100 米以上的距离：这种策略在能见度低的情况下，会增加车辆的反应时间，并可能导致事故。
重要提示：在能见度低时，务必保持安全距离，并注意观察周围车辆，并随时准备采取必要的安全措施。
建议：在能见-低时，应尽量避免高速行驶，并选择相对较平坦的道路行驶。如果必须在能见度低时行驶，应开启危险报警闪光灯，并保持足够的安全距离。

Qwen3-1.7B (non-thinking mode)

在高速公路上遇到**能见度小于 50 米的大雾天气**时，驾驶员应采取以下措施：

正确做法是：
C）开启危险报警闪光灯，并就地停车等待大雾散去。

原因解释：

- **能见度小于 50 米，属于高速公路紧急情况，此时车辆无法安全行驶，必须采取紧急措施。**
- **继续行驶**（如选项 A）会增加事故风险，尤其是在大雾中能见度极低的情况下。
- **减速至 20 公里/小时以下（如选项 B）虽然可以降低风险，但仍可能在雾中发生事故，尤其是当能见度持续低于 50 米时。**
- **保持 100 米以上的距离**（如选项 D）也仅适用于能见度较好时，不能作为大雾天气的应对措施。
- **开启远光灯**（如选项 A）在大雾中更危险，容易造成眩目，反而加剧事故风险。

正确答案：C

Figure 6: A demonstration of output from different models on a Chinese traffic regulation question. Reasoning that is non-compliant with traffic law is labeled as red, while correct reasoning is highlighted as green.

Case Study: Model Outputs on a Physics Optics Question

Question

An object is placed 30 cm in front of a converging (convex) lens with a focal length of 20 cm. Where is the image formed?

A) 60 cm in front of the lens

B) 12 cm behind the lens

C) 60 cm behind the lens

D) 12 cm in front of the lens

Model Responses

openPangu Embedded-1B-V1.1

To determine where the image is formed when an object is placed in front of a converging (convex) lens with a focal length of 20 cm, we can use the **lens formula**:

$$\frac{1}{f} = \frac{1}{v} - \frac{1}{u}$$

where:

- f = focal length of the lens (20 cm for a converging lens)
- u = object distance (-30 cm, since the object is placed in front of the lens and distances are measured from the lens; the negative sign indicates the object is on the left side)
- v = image distance (to be determined)

Step 1: Plug in the known values into the lens formula

$$\frac{1}{20} = \frac{1}{v} - \frac{1}{-30}$$

Simplify the equation:

$$\frac{1}{20} = \frac{1}{v} + \frac{1}{30}$$

Step 2: Solve for $\frac{1}{v}$ Subtract $\frac{1}{30}$ from both sides:

$$\frac{1}{v} = \frac{1}{20} - \frac{1}{30}$$

Find a common denominator (60):

$$\frac{1}{v} = \frac{3}{60} - \frac{2}{60} = \frac{1}{60}$$

Step 3: Solve for v Take the reciprocal of both sides:

$$v = 60 \text{ cm}$$

Step 4: Interpret the sign of v and the image location Since $v = 60$ cm is **positive**, this means the image is **formed on the opposite side of the lens** from the object. Therefore, the image is **formed behind the lens**.

Step 5: Verify using the magnification formula (optional) The magnification m is given by:

$$m = \frac{v}{u}$$

Substitute the values:

$$m = \frac{60}{-30} = -2$$

The negative sign indicates that the image is **inverted** (as expected for a converging lens when the object is placed beyond the focal length). However, the question asks for the **distance of the image from the lens**, which we have determined to be **60 cm** behind the lens. **Final Answer: The image is formed 60 cm behind the lens. Option C is correct.**

$$\boxed{C}$$

Figure 7: A demonstration of output from different models on a physics optics question. Wrong answer is labeled as red, while correct answer is highlighted as green.

gemma-3-1b-it

Here's how to solve this problem:
Understanding Converging Lenses
A converging lens converges (brings together) parallel rays of light. This causes parallel rays to converge at a point called the focal point.
Applying the Lens Equation
The lens equation is: $1/f = 1/d_o + 1/d_i$
Where:

- f = focal length
- d_o = object distance (30 cm)
- d_i = image distance (what we want to find)

Calculations
Set up the equation:
$1/20 = \frac{1}{30} + 1/d_i$
Solve for d_i:
$1/d_i = 1/20 - 1/30$
$1/d_i = (3-2)/60$
$1/d_i = 1/60$
$d_i = 60$ cm
Answer
The image is formed 60 cm behind the lens.
Therefore, the correct answer is B) 12 cm behind the lens.

Qwen3-1.7B (non-thinking mode)

To solve this problem, we use the **thin lens equation**:

$$\frac{1}{f} = \frac{1}{s_o} + \frac{1}{s_i}$$

Where:

- f is the focal length of the lens, - s_o is the object distance from the lens, - s_i is the image distance from the lens.

Given:

- Focal length $f = 20$ cm, - Object distance $s_o = 30$ cm.

Step 1: Plug in the values

$$\frac{1}{20} = \frac{1}{30} + \frac{1}{s_i}$$

Solve for $\frac{1}{s_i}$:

$$\frac{1}{s_i} = \frac{1}{20} - \frac{1}{30} = \frac{3}{60} - \frac{2}{60} = \frac{1}{60}$$

So,

$$s_i = 60 \text{ cm}$$

—

Step 2: Interpret the result

The image is formed **60 cm behind the lens**, which corresponds to option **C)**. — **Correct Answer: C) 60 cm behind the lens.**

Figure 7: A demonstration of output from different models on a physics optics question. (Continued)