

# Youtu-LLM: Unlocking Native Agentic Potential for Lightweight Large Language Models

---

1.96B Parameters | 128K Context Window | Open Source

A lightweight yet powerful language model that harmonizes high computational efficiency with native agentic intelligence through systematic pre-training from scratch.



[github.com/TencentCloudADP/youtu-tip/youtu-llm](https://github.com/TencentCloudADP/youtu-tip/youtu-llm)

Youtu-LLM Team · 2025

# Executive Overview

01

## Introduction & Motivation

The need for lightweight agentic LLMs and the limitations of existing approaches

02

## Pre-Training Data Architecture

General data pipeline and five categories of agentic trajectory construction

03

## Model Architecture & Tokenizer

STEM-oriented tokenizer and dense Multi-Latent Attention design

04

## Multi-Stage Training Recipe

Commonsense-STEM-Agent progressive curriculum across four stages

05

## Post-Training & Alignment

Two-stage SFT and reinforcement learning with verifiers

06

## Evaluation Results & Performance

Comprehensive benchmarks on general and agentic capabilities

### Key Achievement

New state-of-the-art for sub-2B models with native agentic capabilities

**1.96B**

Parameters

**128K**

Context

**200B+**

Agentic Tokens

01

# Introduction & Motivation

## The Need for Lightweight Agentic LLMs

### Core Question

Can lightweight LLMs acquire strong agentic capabilities through pre-training rather than post-augmentation?

### Our Answer

Yes — through systematic agentic pre-training from scratch with innovative data and architecture design.

# The Challenge: Lightweight vs. Performance Trade-off

## ⚠ The Parameter Scaling Problem

State-of-the-art models typically rely on **tens or hundreds of billions of parameters**, incurring substantial computational, financial, and environmental costs during both training and deployment.

**Impact:** These constraints significantly limit accessibility and real-world applicability, especially in latency-sensitive or resource-constrained settings.

## 🧪 Existing Lightweight Approaches

Current methods for improving small models largely rely on **distillation, instruction tuning, or architectural simplification**. Although effective to some extent...

**Limitation:** These methods primarily align output behavior rather than systematically cultivating underlying cognitive capabilities.

## ✖ Resulting Deficiencies

Lightweight models often lack three critical capabilities:

- **Robustness**

Inconsistent performance across diverse scenarios

- **Generalization**

Limited ability to adapt to novel contexts

- **Planning Competence**

Inability to reason through complex multi-step problems

## 🤖 The Agentic Challenge

With complex tasks like **deep research, coding, and tool-augmented workflows**, these limitations become even more pronounced.

**Community Consensus:** Effective agents require intrinsic capabilities for planning, tool execution, state perception, and feedback-driven reflection — not just strong language understanding.

# YouTu-LLM: Core Contributions

## 1 Lightweight Agentic LLM

### Model Achievement

We introduce **YouTu-LLM (1.96B)**, a lightweight open-source language model that significantly outperforms state-of-the-art models of similar or even larger size in agent benchmarks.

Parameter Efficiency

1.96B

Competitive with models 2-4x larger

## 2 Native Agentic Capability Induction

### Training Paradigm Innovation

We propose a **principled training paradigm** that enhances native agentic capabilities through innovations in tokenizer design, data allocation, and multi-stage learning.

⚙️ STEM-oriented tokenizer design

⌚ Strategic data allocation (200B+ agentic tokens)

📦 Multi-stage learning strategy

## 3 Scalable Agentic Trajectory Construction

### Data Engineering Breakthrough

We present **scalable frameworks** for constructing high-quality agentic trajectory data, spanning reasoning, reflection, and planning abilities across multiple domains.

200B

Total Tokens

5

Categories

## 4 Empirical Insights

### Systematic Analysis

We provide the **first systematic analysis** of agentic pre-training in lightweight LLMs, revealing the scalable growth of agentic abilities.

Performance Gain

Average relative improvement across agent benchmarks

+14.4%

02

# Pre-Training Data Architecture

Building the Foundation:  
General Data & Agentic Trajectories

## Total Corpus

10.84T tokens after filtering and processing

## Agentic Data

200B high-quality trajectory tokens

# General Pre-Training Data Pipeline

## Data Collection & Scale

We collected over **10T raw tokens** from various sources, with English as the primary language and Chinese as the secondary focus.

Raw Tokens Collected	<b>10T+</b>
After Filtering	<b>8.7T</b>
Final Training Pool	<b>10.64T</b>

## Corpus Composition

High-quality Chinese and English web pages and encyclopedic knowledge totaled **6.1T tokens**, accounting for over 70% of the raw corpus.

Web & Encyclopedia	<b>6.1T (70%+)</b>
STEM Corpora	<b>700B</b>
Coding Source Data	<b>1,400B</b>
Synthesized Data	<b>500B</b>

## Multi-Dimensional Classification & Filtering

We adopted a solution involving multi-dimensional data classification and quality scoring model for filtering to address uneven quality and domain coverage bias.

**10**

Quality Criteria

**11**

Domain Criteria

**Process:** Semi-automated annotation using Deepseek-R1 with >95% consistency requirement, followed by training a fast classification model based on Qwen3-1.7B.

## Quality Assurance Pipeline

Comprehensive filtering pipeline to ensure high-quality training data:

### 1 Quality Threshold

Average score >8.5 across all 10 criteria

### 2 Heuristic Rules

13-gram duplicate detection, toxicity scoring, code keyword filtering

### 3 De-duplication

MinHash-LSH-based deduplication within length bins

### 4 Decontamination

Aho-Corasick algorithm to minimize test dataset leakage

# Agentic Trajectory Data: Overview

## Trajectory Data Scale & Breakdown

We constructed a massive **200B tokens** of high-quality agentic trajectory data across five distinct categories, each targeting specific agentic capabilities.

### Agentic-CoT Trajectories

Structured thinking with Analysis → Plan → Action → Reflection → Summary

**25B**

12.5%

### Mathematical Trajectories

Atomic ability framework with planning-action-reflection loops

**20B**

10%

### Code Execution Trajectories

Scaling tasks, contexts, and actions with critical action branching

**70B**

35%

### Deep Research Trajectories

Closed-ended (multi-hop QA) and open-ended (report generation) synthesis

**60B**

30%

### Tool-use & Planning

Multi-turn tool-use with planning data via tool graph synthesis

**25B**

12.5%

## Core Design Principles

### Complete Trajectories

Full execution paths with verifiable outcomes

### High Quality

Rigorous filtering and verification processes

### Diverse Coverage

Multiple domains and capability types

### Scalable Construction

Frameworks enabling large-scale synthesis

# Agentic-CoT Trajectory: Structured Thinking

## Motivation & Goal

Reasoning data has progressed through CoT and slow thinking with RL. Current thinking models encapsulate steps in tags, but suffer from **redundancy and repetition**.

**Insight:** By refining raw reasoning content, we can remove inefficient expressions while preserving essential logical structure.

## Agentic-CoT Paradigm

Inspired by agentic workflows, we propose a structured thinking paradigm with **five distinct phases**, each encapsulated in XML-style tags.

### 1 Analysis

Problem breakdown and examination

### 2 Plan

Strategic steps blueprint

### 3 Action

Concrete implementation

### 4 Reflection

Evaluation and error checking

### 5 Summary

Synthesis of findings

## Data Construction Strategy

Our methodology employs a **stratified rewriting approach** using LLMs to systematically generate and refine the dataset:

### 1. Reasoning & Generation

Foundational model processes queries to perform analysis and produce initial responses

### 2. Curation & Extraction

Rigorous filtering and verification, followed by targeted segment extraction

### 3. Synthesis & Assembly

Refined segments intelligently synthesized and concatenated

## Data Statistics & Impact

25B

Total Tokens

4

Domains

**Domains:** Industry, Mathematics, STEM, Coding

**Result:** Agentic-CoT Data significantly enhances agentic thought, leading to improved performance on agent benchmarks.

# Math Trajectory: Atomic Ability Framework

## Motivation & Goal

Mathematical reasoning serves as a cornerstone for evaluating complex reasoning, with objectively verifiable answers and broad difficulty spectrum.

**Insight:** Mathematical trajectories exhibit highly generalizable agentic behavior patterns, naturally capturing planning, execution, and reflection.

## Atomic Ability Hierarchy

We construct a three-level hierarchy decomposing mathematical reasoning into 11 atomic abilities:

### Level 1: Basic Knowledge & Computation

Symbol Recognition | Concept Understanding | Computation

### Level 2: Complex Reasoning & Application

Spatial Perception | Formal Language | Deduction | Construction | Modeling | Theorem Application

### Level 3: Mathematical Meta-cognition

Self-Reflection | Knowledge Acquisition

## Agent Framework Design

We adopt a planning-action-reflection loop that integrates atomic abilities with agent modules:

### Planning Module

Analyzes problem structure → identifies required atomic abilities → decomposes into executable sub-tasks

### Action Stage

Executes each atomic task step-by-step with specified order, inputs, and expected outputs

### Feedback Module

Evaluates problem-solving state → returns feedback signals → enables dynamic adjustment

## Data Statistics

1.38M

Trajectories

20B

Tokens

**Problem Sets:** K12 and mathematics competition level

**Lengths:** Up to 32k and between 32k-128k tokens used separately



**Innovation:** Atomic Thinking framework systematically decouples mathematical reasoning abilities, enabling models to acquire interpretable, stable, and high-quality mathematical reasoning behaviors.

# Code Trajectory: Scaling Three Dimensions

## </> Atomic Code Capabilities

The dataset covers **six core atomic capabilities** essential for autonomous code agents:

### Proactive Exploration & Localization

Autonomously exploring repositories guided by diagnostic logs

### Context-Aware Generation & Completion

Completing critical code segments given repository-level context

### Patch Generation & Editing

Generating precise modifications based on faulty files

### Testing & Verification

Evaluating correctness and localizing failure points

### Environment Comprehension

Interpreting signals from execution environment

### Self-Reflection

Multi-turn self-correction based on environmental feedback

## End-to-End Scaling Strategy

Our approach focuses on **three scaling dimensions**:

### 1. Scaling Tasks

- Multiple open-source sandboxes (SWE-gym, SWE-smith, SWE-rebench)
- Diverse scaffolds (Mini-SWE-agent, SWE-gym, OpenHands, R2E-gym)
- Extended domains: data science, front-end, game development

### 2. Scaling Contexts

- Static evaluation for long-tail repositories without executable images
- Tasks: fault localization, refactoring (highly scalable & verifiable)
- Fosters proactive exploration and dynamic comprehension

## Data Statistics

70B

Total Tokens

35%

Of Agentic Data

**Key Innovation:** Critical Actions (Editing, Testing) with trajectory branching for both successful and failed paths

# Deep Research Trajectory: Dual-Pronged Synthesis

## 🔍 Motivation & Task Categories

Deep Research represents a leap from simple retrieval to complex, self-directed knowledge discovery.

### Closed-ended Tasks

Multi-hop QA with well-defined, verifiable answers

### Open-ended Tasks

Comprehensive research reports or scientific papers

## ❓ Closed-ended Synthesis

Multi-hop QA trajectory synthesis with three key steps:

### 1 QA Generation

Entity-knowledge pairs from diverse sources with LLM-as-a-judge scoring

### 2 Trajectory Diversification

Multiple frameworks + search perturbations (source masking, top-K suppression)

### 3 Failed Trajectory Utilization

Append analysis segment with step-by-step examination of incorrect responses

## 📄 Open-ended Synthesis

Dual-pronged strategy for comprehensive report generation:

### Forward Trajectory Synthesis

- **Question Acquisition:** Encyclopedia, academic, financial sources
- **"Think Twice" Framework:** Systematic reconsideration before report generation
- **Report Generation:** Consolidate → Outline → Develop → Refine

### Inverse Trajectory Synthesis

- **Advantage:** Leverage existing high-quality content (papers, reports)
- **Method:** Reconstruct search paths backwards using citation graphs
- **Result:** 20B tokens with average length ~32K

## 📊 Data Statistics & Atomic Capabilities

60B

Total Tokens

30%

Of Agentic Data

### Atomic DR Capabilities:

Plan Self-Reflection Summary Reading

Explicit reinforcement of individual skills builds robust foundation for complex research.

# Tool-use & Planning Trajectory

## Motivation & Goal

Tool-use and strategic planning ability determines how the LLM interacts with the environment, moving beyond static internal reasoning to [dynamic, goal-oriented problem solving](#).

**Scope:** We compiled 25B tokens of tool invocation and task planning trajectory data not limited to specific domains.

## Step 1: Atomic Tool Collection

Constructed mixed skill library from [three perspectives](#):

### Breadth

Thousands of diverse tools: APIs, MCPs, protocols

### Depth

Simple (weather, date) to professional (text2sql, code interpreters)

### Dependencies

Direct dependencies between skills (retrieve → search → modify)

## Step 2: Multi-turn Trajectory Synthesis

Based on the tool graph, we synthesized multi-turn tool-use trajectory data:

### 1 Random Walk

Generate valid tool execution chain conforming to dependencies

### 2 Adversarial Generation

Two LLMs as User & Assistant simulate multi-turn trajectories

### 3 Planning Generation

Expert LLM summarizes trajectory and generates planning data

**User Side Rules:** Vague statements, repeated requests, spelling errors, personality simulation

## Steps 3 & 4: Quality & Negative Samples

### Data Quality Verification (3-Step)

- Format checking (JSON/XML, tool existence, parameters)
- Tool call feasibility (local re-execution)
- LLM comprehensive quality judgment

### Negative Trajectory Augmentation

5% selected for negative augmentation: delete tools, modify statements, insert topic shifts



**Result:** 25B tokens of high-quality tool-use and planning trajectory data enabling robust general-purpose tool-use capabilities.

03

# Model Architecture & Tokenizer

Technical Foundation for  
Lightweight Performance

## Architecture

Dense Multi-Latent Attention (MLA)

## Tokenizer

STEM-oriented multi-stage training

# STEM-Oriented Tokenizer Design

## </> Pre-Tokenization Scheme

A **stricter pre-tokenization** design preventing tokens from spanning unrelated semantic units:

### CJK Handling

Chinese, Japanese kana, Hangul, CJK punctuation → standalone units

### English Design

GPT-4o style: single punctuation or space-prefixed capital + lowercase sequence

### Numeric Tokenization

Only atomic digit tokens 0-9 (no multi-digit tokens)

## Ξ Performance Comparison

### Compression Rate (General)

vs Llama3 (1.00 baseline)

**1.15x**

### Compression Rate (Reasoning)

-10% improvement on STEM data

**1.10x**



**Final Spec:** Byte-level BPE · Vocab Size: 128,256 · Multi-stage training balances basic language ability with domain-specific reasoning capacity.

**128k**  
Vocab Size



## Multi-Stage Training Strategy

Progressive vocabulary construction with **four distinct stages**:

### 1 Base Vocabulary

~200k base → 101k tokens (100k ASCII + 1k multilingual)

### 2 Chinese Tokens

Rebalanced corpus → 121k tokens (removed >4 char tokens)

### 3 Reasoning Tokens

+4k code +3k math/technical → 128k total (256 reserved)

## 🏆 Key Benefits

### ✓ 5% Efficiency Gain

On general pre-training data vs Qwen3

### ✓ 10% on Reasoning Data

Gain increases on STEM-focused corpora

### ✓ Best Performance

On Commonsense and STEM tasks in controlled experiments

# Dense Multi-Latent Attention Architecture

## Architecture Choice: Dense MLA

As a lightweight LLM for [on-device scenarios](#), we adopt dense Multi-Latent Attention (MLA) from the beginning.

### Why not MoE?

Mixture-of-Experts doesn't offer significant speed advantages for on-device due to frequent I/O operations.

### MLA Advantages

- Low-rank compression of KV Cache
- Larger intermediate projection matrices
- Improved expressiveness with constrained parameters
- Surpasses vanilla GQA in attention mechanism

## 实验验证

We compared GQA and MLA by pre-training 1B-parameter models from scratch on 500B tokens:

### GQA-1B

Wiki PPL: 6.0

### MLA-1B

Wiki PPL: 7.2

**Result:** MLA-1B outperformed GQA in all formats and languages tested.

## YouTu-LLM Configuration

**1.96B**

Parameters

**32**

Layers

**2,048**

Hidden Size

**16/16**

Q/KV Heads

**128K**

Max Length

**128K**

Vocab Size

## MLA Configuration

Consistent with DeepSeek-V3: KV lora rank 512, Q lora rank 1,536, QK nope head 128, QK rope head 64, V head 128

## Architecture Comparison

**Qwen3-1.7B**

GQA

**SmoILM3-3B**

GQA

**MiniCPM3-4B**

MLA

**YouTu-LLM 2B**

MLA ✓

## MULTI-STAGE TRAINING RECIPE

# Four-Stage Training Pipeline

## 1 Commonsense Pre-training Foundation Stage

Tokens	8.16T
Sequence Length	8,192
Web/Encyclopedia	75%
Warmup Steps	2,000

## 3 General Mid-training Context Extension & Decay

Context Length	8k → 32k → 128k
Learning Rate	4e-4 → 4e-5

**Outcome:** Gains stable STEM, coding, and long-context capabilities



**Total Training:** 10.84T tokens following "Commonsense-STEM-Agent" design principle, aligning with Bruner's Spiral Curriculum theory and Ausubel's Progressive Differentiation principle.

## 2 STEM & Coding-centric Capability Development

Data Shift	STEM/Coding ↑60%
Learning Rate	Max (4e-4)

**Focus:** Systematically increase STEM and coding data proportion to build core reasoning capabilities

## 4 Agentic Mid-training Agentic Capability Induction

Agentic Data	↑60%
Learning Rate	→ 1e-7

**Key Insight:** Agentic training after long-context yields larger gains – model better captures cross-segment information

04

# Post-Training & Alignment

From Base Model to  
Instruction-Following Assistant

## SFT Approach

Two-stage strategy with reasoning-first

## RL Framework

Multi-task with verifiers & consistent sampling

# Supervised Fine-Tuning: Two-Stage Strategy

## Data Engineering Pipeline

We curated a comprehensive dataset across 10 categories :

- |                         |                          |
|-------------------------|--------------------------|
| 1. Mathematics          | 2. Code                  |
| 3. Scientific Reasoning | 4. Agentic Data          |
| 5. General Knowledge QA | 6. Instruction Following |
| 7. Role-Playing         | 8. Creative Writing      |
| 9. Multi-turn Dialogue  | 10. Safety & Alignment   |

## Reasoning Answer Construction

Two scenarios for explicit long CoT reasoning:

### Queries without Answers

Use advanced LLMs to generate comprehensive responses with reasoning + final answer

### Queries with Answers

High-quality: reconstruct CoT; Low-quality: discard & regenerate entire response

## Multi-Stage Data Cleaning

### 1. Heuristic Filtering

- Truncation errors (mid-sentence termination)
- Mixed-language noise (garbled encodings)
- Repetition loops (pathological phrase repetition)

### 2. Model-Based Quality Scoring

Teacher models score on coherence, fluency, and accuracy. Samples below threshold discarded.

### 3. De-contamination

32-gram exact matching to identify and remove training-test overlap

## Two-Stage SFT Strategy

### Stage I: Reasoning SFT

Logic-dense dataset to "cold-start" reasoning engines:

Math 40%    Code 30%    Science 20%    Agentic 10%

### Stage II: General SFT

Full data diversity + Stage I subsets to prevent catastrophic forgetting. Dual-mode capability:

Think Mode    Non-Think Mode

# Reinforcement Learning: Tasks & Training Dynamics

## Tasks & Verifiers

### Mathematical Reasoning

- Competition-level problems (elementary to advanced)
- Rigorous deduplication using embedding similarity
- Structured formats for automated validation
- Excluded multiple-choice to prevent reward hacking
- Augmented with practical math reasoning

### Code Generation

- Primary: Autonomous code generation + execution validation
- Secondary: Program execution simulation (I/O mapping)
- Cross-reference standard solutions with test suites

### Complex Instructions

100+ foundational instructions with hybrid rewards: rule-based for simple tasks, LLM-as-judge for complex.

## ⚠️ BF16 vs FP16: Critical Finding

BF16 precision shows progressive divergence between training and inference. After ~50 steps, performance drops significantly.

## Safety & General Rewards

### Safety Design

Informative guidance over categorical refusal:

E.g., for bomb making queries → explain legal/safety risks and pivot to constructive topics

### General Rewards

- **Thinking Format Preservation:** Rule-based verification of markers
- **Language Consistency:** Pre-label target language, penalize drift
- **Repetition Detection:** Monitor n-gram frequencies, suppress cyclic patterns

## ⌚ Training Dynamics

### FP16 Precision

More effective than BF16 at maintaining training-inference consistency. Reduces numerical drift for precise gradient updates.

### Consistent Sampling (CF-GRPO)

Filter groups with  $K(q) < \tau$  (0.01). Model only learns from prompts where policy drift is constrained.

# Evaluation Results

Comprehensive Performance  
Analysis

## Evaluation Scope

General & agentic benchmarks on base and instruct models

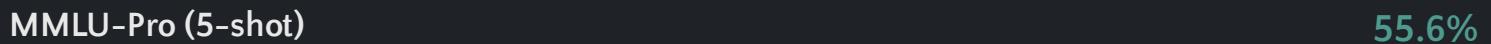
## Key Achievement

SOTA for sub-2B models, competitive with larger models

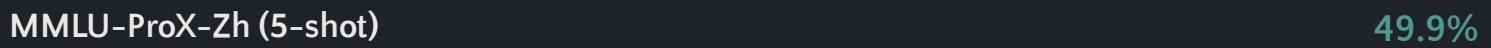
## EVALUATION RESULTS

# Base Model: General Capabilities

### Commonsense Knowledge

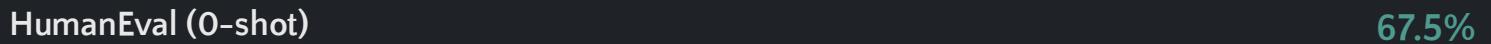


vs Qwen3-1.7B: 34.9%, SmoLLM3-3B: 38.1%

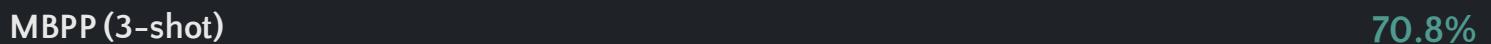


Chinese subset performance

### Coding Proficiency

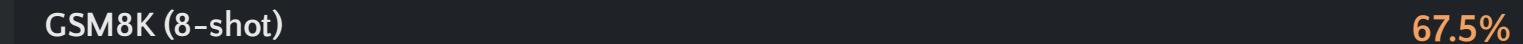


Functional code generation

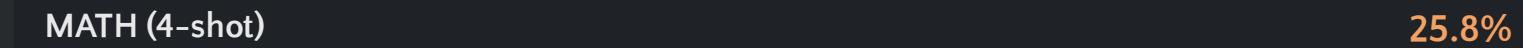


Basic programming tasks

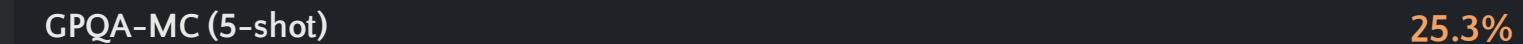
### STEM Capabilities



Grade-school math (vs Qwen3-1.7B: 28.1%)

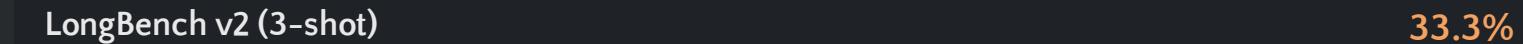


Competition-level (vs Qwen3-1.7B: 3.1%)

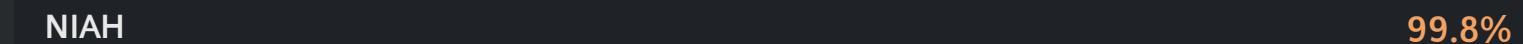


Expert-level scientific QA

### Long Context



Realistic long-context multitasks



Needle in a haystack recall

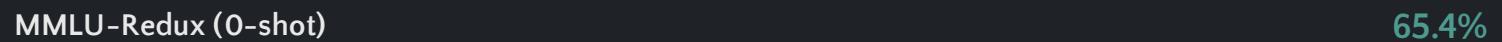


**Key Achievement:** Youtu-LLM 2B Base significantly outperforms similar-sized baselines and achieves competitive results with larger Qwen3-4B Base.

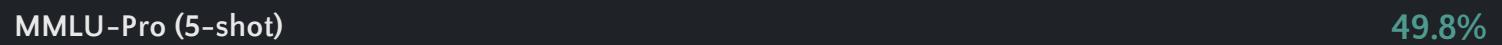
**SOTA**  
Sub-2B

# Instruct Model: Comprehensive Performance

## Commonsense Knowledge

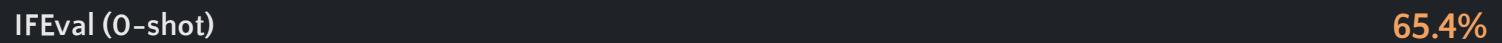


vs Qwen3-1.7B: 29.4%, SmolLM3-3B: 41.3%

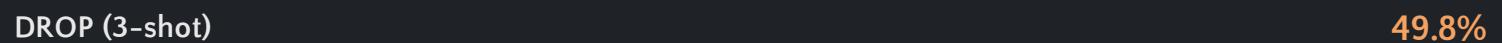


vs Qwen3-1.7B: 30.2%, DeepSeek-R1-Distill-Llama-8B: 44.2%

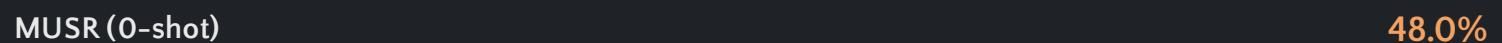
## Instruction Following & Reasoning



Objective instruction following

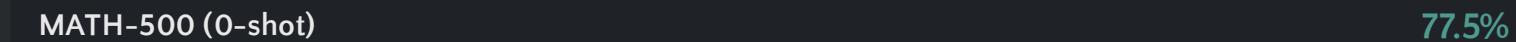


Discrete reasoning over paragraphs

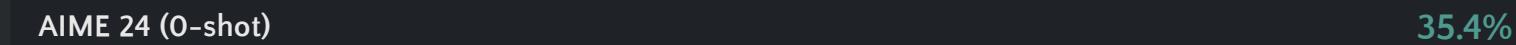


Long-chain logical deduction

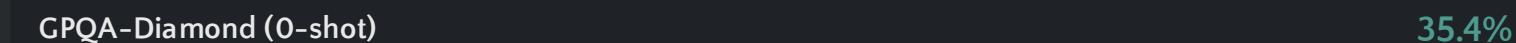
## STEM Capabilities



Competition-level mathematics

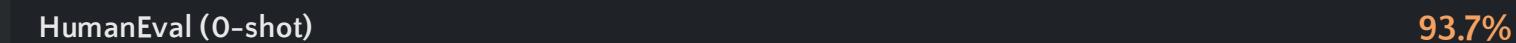


Elite-level problem solving

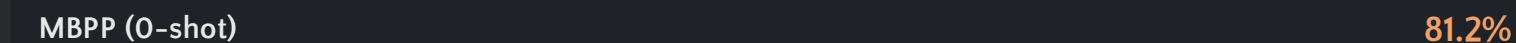


Expert-level scientific knowledge

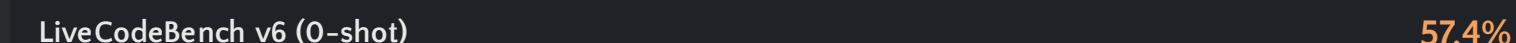
## Coding Proficiency



vs DeepSeek-R1-Distill-Qwen-1.5B: 84.8%



vs Qwen3-1.7B: 53.0%



Latest competitive programming problems

## EVALUATION RESULTS

# Agentic Evaluation: Base Model

## APTBench: Agentic Potential Evaluation

APTBench serves as a benchmark specifically designed to evaluate agent capabilities of base models. It adopts a combination of few-shot multiple-choice question (MCQ), text completion (TC) and true/false questions, making it suitable for efficient assessment during pre-training.

4

Domains

MCQ

Format

TC

Format

T/F

Format

### </>Code Domain

Planning: Stepwise environment setup strategies or SWE debugging plans (MCQ)

Action: Execution of terminal commands based on prior context (TC)

Atomic: Error handling, bug localization, fix/test patch identification (MCQ)

### 🔍 Deep Research Domain

Planning: Stepwise decision-making for information seeking/organizing (MCQ)

Action: Synthesizes gathered information into outputs from short answers to detailed reports (TC)

Atomic: Checks citation accuracy and factual grounding of statements (MCQ)

## 🏆 Performance Results

Model

Youtu-LLM 2B

Code

37.9%

DR

38.6%

Math

68.0%

Tool

64.2%

**Key Finding:** Youtu-LLM 2B Base achieves 37.9% overall, close to Qwen3 4B Base (41.9%), significantly outperforming other similar-sized models.

## EVALUATION RESULTS

# Agentic Evaluation: Instruct Model

### 🔍 Deep Research

GAIA

33.9%

vs Qwen3-1.7B: 11.4%

xbench

19.5%

vs Qwen3-1.7B: 11.7%

### UserCode Agent

SWE-Bench-Verified

17.7%

vs Qwen3-1.7B: 0.6%

EnConda-Bench

21.5%

Environment setup tasks

### 🛠 Tool Use

BFCL V3

58.0%

vs Qwen3-1.7B: 55.5%

τ2-Bench

15.0%

vs Qwen3-1.7B: 2.6%

### 📈 Performance Summary

#### Key Achievement

Youtu-LLM 2B achieves superior performance compared to the best baseline on most benchmarks, with significant improvements.

#### Notable Limitations

- No suitable agentic mathematical benchmark exists for instruct models
- Gemma3 4B and Llama3.1 8B lack native tool-calling mechanism

**Evaluation Protocol:** All models evaluated in thinking mode (generating explicit thought processes) with temperature = 1.0, top-p = 0.95, top-k = 20, presence penalty = 1.5, max output 32,768 tokens.

# Agentic Mid-Training Analysis

## ↳ Mid-Training Scaling Behavior

We observe a [clear logarithmic trend](#) between training tokens and model agentic performance:

### Key Observations

- Significant improvement within first 34B tokens (~20% of total)
- Consistent performance gains across entire 340B-token budget
- Overall improvement of more than 6% on APTBench
- Logarithmic fit confirms scalable growth pattern

**34B**

First Milestone

**+6%**

Total Gain

## 💡 With vs Without AMT Comparison

Performance comparison of post-trained models with and without agentic mid-training (AMT):

GAIA	+9.0%
xbench	+8.3%
SWE-Bench-Verified	+42.7%
EnConda-Bench	+8.6%
τ2-Bench	+20.0%

## ↳ Pass@k Analysis on SWE-Bench

AMT yields consistent and substantial performance uplift across all k values:

Pass@1	17.7% vs 12.4%
42.7% relative improvement	

Pass@8	33.4% vs 26.2%
Margin widens as k increases	

**Insight:** 95% confidence intervals remain largely non-overlapping, signifying statistically significant gains.

# Key Insights & Future Directions

Implications and Research

Roadmap Ahead

## Main Insights

Lightweight models can possess strong intrinsic agentic capabilities

## Future Work

World models, efficient architectures, multimodal extension

# YouTu-LLM: Unlocking Native Agentic Potential in Lightweight Models

## ✓ Key Achievements

- New state-of-the-art for sub-2B models with native agentic capabilities
- Systematic agentic pre-training from scratch (1.96B parameters)
- 200B tokens of high-quality agentic trajectory data across 5 categories
- Competitive performance with models 2-4x larger

## 👉 Future Directions

- Evolve into world model simulating execution dynamics
- Explore efficient architectures (Diffusion LLMs) for inference speed
- Extend to multimodal scenarios with native omni-modal perception
- Bridge gap with large proprietary LLMs through systematic advancement

**Core Insight:** Lightweight models can possess strong **intrinsic agentic capabilities** through systematic pre-training – computational efficiency and deep reasoning can coexist effectively.