Mastering DeepScaleR

Build and deploy AI Models with Ollama

Riddhi Airan

• what is DeepScaleR ?

- Fine-tuned version of DeepSeek-R1-Distilled-Qwen-1.5B
- specifically designed to surpass open AI's O1 preview model in mathematical reasoning tasks.
- High accuracy and efficiency
- Extremely lightweight yet powerful model for AI-driven computations.

• why Choose DeepScaleR ?

- Lightweight (1.5B parameters) yet powerful
- Outperforms larger models in structured math evaluations.
- Runs locally with Ollama (No API Costs!)
- Supports fine-tuning & Customization
- Low-latency inference (ideal for real-time applications)

Riddhi Airan

• Overview of DeepscaleR's Capabilities

1. Distilled from DeepSeek-R1 for Efficiency.

- DeepSeek-R1 is a family of open-source LLMs with strong performance in text generation and reasoning.
- DeepScaleR refines this architecture to be optimized for math-heavy and structured problem-solving tasks.
- It retains the speed and efficiency of DeepSeek-R1 while enhancing accuracy for numerical and symbolic reasoning.

2. Advanced Mathematical Capabilities

- Optimized for symbolic, algebraic, and numerical tasks, including :
  - Arithmetic and Algebra { +, −, *, / and algebraic equations }
  - Linear Algebra { matrix operations, determinants and agent values }
  - Calculus { Derivatives, Integrals, limits and series expansion }
  - Probability and statistics { Bayesian analysis, hypothesis testing
  - Logical and Symbolic Reasoning { Theorem proving, formal logic }

3. Small yet powerful (1.5B Parameters)

- DeepScaleR is significantly smaller than most mainstream LLMs yet delivers comparable (or better) performance in its target areas.
- Designed for low-latency inference, making it ideal for edge devices, personal AI assistants, and embedded AI Systems.
- Supports quantization techniques to further reduce memory footprint while retaining accuracy.

4. Optimized for Deployment

- Compatible with ollama, enabling fast and easy local inference.
- Can run on CPUs and GPUs, making it more accessible than larger models.
- Supports fine-tuning with LoRA and QLoRA, allowing easy customization for specific domains

Riddhi Airan

Comparison :

DeepScaleR Vs. OpenAI's 01 - Preview

Feature	DeepScaleR (1.5B)	OpenAI's 01 - preview
Model Size	1.5B Parameters	9B + Parameters
Performance on Math tasks	Better in structured math evaluations (algebra, calculus, linear algebra)	Strong in general problem-solving but weaker in structured math.
Inference speed	Fast (optimized for low-latency inference)	Slower due to larger model size
Resource Requirement	lightweight, runs on CPU and GPU	Heavier, requires GPU acceleration
Fine-Tuning capability	LoRA/QLoRA supported for domain specific improvements	limited fine-tuning options
Deployment Ease	Runs with Ollama, easy local setup	Requires cloud-based APIs
Licence & Accessibility	Open-source & locally deployable	Closed-source, API-only

Key takeway

DeepScaleR Outperforms open AI - 01 - preview in mathematical reasoning, has a much smaller footprint and is ideal for local AI deployments.

Riddhi Airan

• Performance On Math Evaluations

• Benchmarking against other models

– DeepScaleR was tested on multiple math reasoning Datasets:

- GSM8K (Grade School Math 8K) → arithemetic & word problems
- MATH Dataset → High School and Olympiad level math.
- Hendrycks MATH → complex math reasoning
- APPS (AI problem solving) → Algorithm and problem solving tasks.

Model	GSM8K (Accuracy %)	MATH Dataset	Inference speed
Deepscaler (1.5B)	85 %	78 %	Fast (CPU + GPU)
Open AI's o1-preview	80 %	70 %	Slower (GPU)
GPT-3.5-turbo	82 %	74 %	Medium
Mistral - 7B	79 %	71 %	Slow (GPU)

– Key Insights from Performance Evaluations

- DeepScaleR outperforms Open AI's o1-preview on structured math datasets.
- Achieves near GPT 3.5 levels of accuracy in problem-solving with only 1.5B parameters.
- Much faster inference speed compared to larger models

Riddhi Airan

• What is Ollama ?

- open-source framework that enables easy local deployment and inference of large language Models.
- Designed for efficiency, simplicity, and fast API model execution on personal computers, workstations, and servers.
- Download, run, and interact with models like DeepScaleR, Mistral, Llama, and more without needing complex Cloud based APIs and GPU Clusters.

• Introduction to the Ollama Framework

• What Does Ollama do ?

- Runs LLMs locally without requiring cloud-based APIs.
- Supports Optimized models like DeepScaleR, Mistral, Qwen, Llama and other open-source LLMs.
- Provides a simple interface to interact with models via CLI, API, or Python SDK
- Works efficiently on both CPU and GPU, making AI accessible on a variety of machines.

• Key Features of ollama

- Local AI Execution - No external API Calls & runs entirely on our machine.
- Lightweight & Optimized - uses quantized models for efficiency.
- Model Management - can easily pull, update and switch between models.
- Build-in API - offers a simple way to integrate AI into applications.
- Secure & private - No data leaves your local environment.

• How Does Ollama Work ?

- Ollama provides a unified way to interact with different LLMs using a simple Command line Interface [CLI] or API
- Download & Run Models
- Interact via API

Riddhi Airan

• Why use Ollama for AI Inference?

• **No More Expensive API Calls:** Instead of relaying on cloud based APIs, like open AI or anthropic that charge per request. Ollama allows to run AI models locally, saving cost and ensuring unlimited usage.

• **Faster Response Times:** It eliminates the latency issue of cloud hosted models, making it ideal for real time applications like Chatbots, assistants, and AI driven automation.

• **Works Offline:** It does not require an internet connection, making it perfect for :

- Edge AI Applications (AI running on local devices)
- Privacy-focused AI (no cloud data transmission)
- AI for secure Environments (finance, defence, healthcare, etc)

• **Supports Multiple Models:** Ollama supports multiple LLM architectures, so can switch models on demand.

- Ollama run deepscaleR (for math)
- Ollama run mistral (for general reasoning)
- Ollama run qwen (for multilingual tasks)

• **Optimized for CPUs and GPUs**

- Runs efficiently on consumer grade hardware
- Supports quantization (like GGML, GPTQ) for lower memory usage and better speed.
- No need for high-end GPUs, but scales well with them

Riddhi Airan

• How Ollama Simplifies LLM Deployment

• Easy Installation & setup

- Unlike traditional AI Development, which requires python environments, Cuda, and dependency management.

• Install ollama (Mac / linux / windows WSL)

- Go to Ollama.com
- Download on your machine

• Pull a model

- Go to terminal
- check version : Ollama -v
- pull model: Ollama pull deepscaler

• Run a model locally

- run model : Ollama run deepscaler
- chat with it
- exit model : /bye

• Built-in Model Serving

- Ollama includes a local API server, so you don't need to set up Flask, FastAPI, or other servers separately.

- Seamless Integration with Applications :

• Python SDK • Java Script API • Dockerized AI Apps

• Why use Ollama for DeepscaleR ?

- No Cloud Costs : Runs entirely on your local Machine
- Fast AI Inference : low latency, ideal for real time AI
- Easy Deployment : CLI and API make model serving effortless
- Supports multiple models : Switching between AI models easily
- Private & Secure : No data leaves your local system

Riddhi Airan

Hands - On Project

1. Setting Up the Environment

a. Install ollama
b Download and Run DeepscaleR locally
c. Running inference with DeepscaleR

2. Project 1 : AI - Powered Math Solver

- Use DeepscaleR to build an AI-powered math solver that can handle complex math problems such as algebra, calculus, and linear algebra.
- Step by step :

- Using DeepScaler for solving mathematical equations
- Implementing a web-based math assistant using Python & Gradio.

3. Project 2 : AI Chatbot using DeepScaleR

- Handle general-purpose conversations, and we will deploy it as an API while ensuring low-latency responses
- Step 1 : Building a chatbot for General - Purpose Conversation
- Step 2 : Deploying DeepScaleR in an API
- Step 3 : Building a web-based Chatbot UI with Gradio

Riddhi Airan